REFERENCE DU LANGAGE DES COMPOSANTS ACTIONSCRIPT™ 2.0

Adobe

Référence du langage des composants ActionScript™ 2.0

Table des matières

CHAPITRE 1

Composants ActionScript 2.0

1

La *Référence du langage des composants ActionScript 2.0* détaille chaque composant qui est disponible dans la version 2 de l'architecture des composants Adobe avec son interface de programmation d'applications (API). Pour savoir comment utiliser, personnaliser et créer des composants de la version 2, consultez le guide *Utilisation des composants ActionScript 2*.

REMARQUE	Adobe Flash CS3 Professional englobe à la fois les composants d'ActionScript 2.0 et d'ActionScript 3.0. Toutefois, vous ne pouvez pas mélanger ces deux groupes de composants. Vous devez définir l'un ou l'autre pour une application donnée. Flash CS3 présente soit des composants ActionScript 2.0, soit des composants ActionScript 3.0 selon que vous avez ouvert un fichier ActionScript 2.0 ou ActionScript 3.0. Lorsque vous créez un nouveau document Flash CS3, vous devez spécifier soit un fichier Flash (ActionScript 3.0), soit un fichier Flash (ActionScript 2.0). Lorsque vous ouvrez un document existant, Flash examine les Paramètres de publication afin de déterminer quel ensemble de composants utiliser. Pour plus d'informations sur les composants ActionScript 3.0, consultez *Utilisation des composants ActionScript 3.0*.

Dans cet ouvrage, chaque description de composant contient des informations sur les thèmes suivants :

- Interaction clavier
- Aperçu en direct
- Accessibilité
- Définition des paramètres du composant
- Utilisation du composant dans une application
- Personnalisation du composant avec des styles et des enveloppes
- Méthodes, propriétés et événements ActionScript

Les composants sont présentés par ordre alphabétique. Vous les trouverez également classés par catégorie dans les tableaux qui suivent.

Ce chapitre contient les sections suivantes :

Types de composants

Les tableaux suivants répertorient les composants de la version 2, classés par catégorie.

Composants d'interface utilisateur (IU)

Composant	Description
Composant Accordion	Jeu d'affichages verticaux se chevauchant, dont les boutons supérieurs permettent aux utilisateurs de passer d'un affichage à l'autre.
Composant Alert	Fenêtre contenant un message et des boutons pour capturer la réponse de l'utilisateur.
Composant Button	Bouton pouvant être redimensionné et personnalisé à l'aide d'une icône.
Composant CheckBox	Permet aux utilisateurs de faire un choix booléen (true ou false).
Composant ComboBox	Permet aux utilisateurs de choisir une option dans une liste déroulante. Ce composant peut contenir un champ de texte susceptible d'être sélectionné en haut de la liste et qui permet aux utilisateurs d'effectuer une recherche dans la liste.
Composant DataGrid	Permet aux utilisateurs d'afficher et de manipuler plusieurs colonnes de données.
Composant DateChooser	Permet aux utilisateurs de sélectionner une ou plusieurs dates dans un calendrier.
Composant DateField	Champ de texte non sélectionnable, avec icône de calendrier. Lorsque l'utilisateur clique dans le cadre de sélection du composant, Flash affiche un composant DateChooser.
Composant Label	Champ de texte d'une ligne non modifiable.
Composant List	Permet aux utilisateurs de choisir une ou plusieurs options dans une liste déroulante.
Composant Loader	Conteneur détenant un fichier SWF ou JPEG chargé.
Composant Menu	Menu standard d'application bureautique qui permet aux utilisateurs de choisir une commande dans une liste.
Composant MenuBar	Barre de menus horizontale.
Composant NumericStepper	Zone de texte équipée de flèches permettant d'augmenter et de réduire la valeur d'un nombre.

Composant	Description
Composant ProgressBar	Présente la progression d'un processus, par exemple une opération de chargement.
Composant RadioButton	Permet aux utilisateurs de choisir parmi des options qui s'excluent réciproquement.
Composant ScrollPane	Affiche des clips, des bitmaps et des fichiers SWF dans une zone délimitée à l'aide de barres de défilement automatiques.
Composant TextArea	Champ de texte à plusieurs lignes modifiable.
Composant TextInput	Champ de saisie de texte à une ligne modifiable.
Composant Tree	Permet à un utilisateur de manipuler des informations hiérarchisées.
Composant Window	Fenêtre contenant une barre de titre, une légende, une bordure, un bouton Fermer et présentant du contenu à l'utilisateur.
Composant UIScrollBar	Permet d'ajouter une barre de défilement à un champ de texte.

Gestion des données

Composant	Description
Classes de liaison des données	Classes qui implémentent la fonctionnalité de liaison des données Flash lors de l'exécution.
Composant DataHolder	Contient des données et peut être utilisé en tant que connecteur entre composants.
API DataProvider	Modèle pour les listes de données à accès linéaire. Il offre des capacités de manipulation simple de tableaux qui distribuent leurs modifications.
Composant DataSet	Bloc de construction pour la création d'applications orientées données.
Composant RDBMSResolver	Permet d'enregistrer les données sur toute source de données prise en charge. Ce composant traduit le format XML qui peut être reçu et analysé par un service Web, JavaBean, un servlet ou une page ASP.
Classes de service Web	Classes permettant d'accéder aux services Web qui utilisent le protocole SOAP (Simple Object Access Protocol). Ces classes se trouvent dans le package mx.services.
Composant WebServiceConnector	Fournit un accès sans script aux appels de méthode de service Web.

Composant	Description
Composant XMLConnector	Lit et rédige des documents XML à l'aide des méthodes HTTP `GET` et `POST`.
Composant XUpdateResolver	Permet d'enregistrer les données sur toute source de données prise en charge. Ce composant traduit DeltaPacket en XUpdate.

Composants de support

Composant	Description
Composant FLVPlayback	Permet d'inclure un lecteur vidéo dans votre application Flash afin de lire de la vidéo progressive en continu sur HTTP depuis un service de diffusion en continu des vidéos de Flash (FVSS, Flash Video Streaming Service) ou FMS (Flash Media server).
Composant MediaController	Contrôle la lecture de support en flux continu dans une application (reportez-vous à « Composants média », à la page 865).
Composant MediaDisplay	Affiche le support en flux continu dans une application (reportez-vous à « Composants média », à la page 865).
Composant MediaPlayback	Combinaison des composants MediaDisplay et MediaController (reportez-vous à « Composants média », à la page 865).

Gestionnaires

Classe	Description
Classe DepthManager	Gère la profondeur des objets dans les piles.
Classe FocusManager	Gère la navigation entre les composants via la touche de tabulation. Gère également les changements de focus lorsque l'utilisateur clique dans l'application.
Classe PopUpManager	Permet de créer et de supprimer des fenêtres contextuelles.
Classe StyleManager	Permet d'enregistrer les styles et de gérer les styles hérités.
Classe SystemManager	Permet de choisir la fenêtre de niveau supérieur qui est activée.
Classe TransitionManager	Permet de gérer les effets animés sur les diapositives et les clips.

Ecrans

Catégorie	Description
Classe Form	Permet de manipuler les écrans d'applications de formulaires lors de l'exécution.
Classe Screen	Classe de base des classes Slide et Form.
Classe Slide	Permet de manipuler les écrans de diaporamas lors de l'exécution.

Autres listes dans cet ouvrage

Ce guide décrit également plusieurs classes et API qui ne sont pas incluses dans les catégories de composants répertoriées dans la section précédente. Ces classes et API sont répertoriées dans le tableau suivant.

Elément	Description
API CellRenderer	Ensemble de propriétés et de méthodes que les composants à base de listes (List, DataGrid, Tree, Menu et ComboBox) utilisent pour manipuler et afficher le contenu de cellules personnalisées pour chacune de leurs lignes.
Interface Collection	Permet de gérer un groupe d'éléments apparentés, appelés éléments de collection. Chaque élément de collection de cet ensemble présente des propriétés décrites dans les métadonnées de sa définition.
Classe DataGridColumn	Permet de créer des objets à utiliser comme colonnes d'une grille de données.
Classe Delegate	Permet d'exécuter une fonction transmise d'un objet à un autre dans le contexte du premier objet.
Interface Delta	Permet d'accéder à l'objet transféré, à la collection et aux modifications apportées à l'objet transféré.
Classe DeltaItem	Fournit des informations sur une opération individuelle effectuée sur un objet de transfert.
Interface DeltaPacket	Avec l'interface Delta et la classe DeltaItem, permet de gérer les modifications apportées aux données.
Classe EventDispatcher	Permet d'ajouter et de supprimer des écouteurs d'événements de sorte que votre code puisse réagir de façon appropriée à un événement.
Interface Iterator	Permet de parcourir les objets d'une collection.

Elément	Description
Classe MenuDataProvider	Permet aux occurrences XML affectées à une propriété `Menu.dataProvider` d'utiliser des méthodes et des propriétés pour manipuler leurs propres données et les menus associés.
Classe RectBorder	Décrit les styles utilisés pour contrôler les bordures des composants.
Classe SimpleButton	Permet de contrôler certains aspects de l'apparence et du comportement d'un bouton.
Interface TransferObject	Définit un ensemble de méthodes que les éléments gérés par le composant DataSet doivent implémenter.
Interface TreeDataProvider	Ensemble de propriétés et de méthodes utilisées pour créer du code XML pour la propriété `Tree.dataProvider`.
Classe Tween	Permet d'utiliser ActionScript pour déplacer, redimensionner et appliquer aisément des fondus aux clips sur la scène.
Classe UIComponent	Fournit des méthodes, des propriétés et des événements qui permettent aux composants de partager certains comportements communs.
Classe UIEventDispatcher	Permet aux composants de déclencher certains événements. Cette classe est combinée à la classe UIComponent.
Classe UIObject	Classe de base de tous les composants.

CHAPITRE 2

Composant Accordion

2

Il s'agit d'un navigateur contenant une série d'enfants qui s'affichent l'un après l'autre. Les enfants doivent être des objets héritant de la classe UIObject (qui inclut tous les composants) ; ces enfants forment généralement une sous-classe de la classe View. Il s'agit notamment de clips affectés à la classe mx.core.View. Pour conserver l'ordre de tabulation dans les enfants d'un accordéon, les enfants doivent également être des occurrences de la classe View.

REMARQUE

Le composant Accordion est pris en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Un accordéon crée et gère les boutons d'en-tête sur lesquels un utilisateur peut cliquer pour passer d'un enfant de l'accordéon à un autre. La disposition d'un accordéon est verticale et l'accordéon est doté de boutons d'en-tête qui couvrent toute sa largeur. Un en-tête est associé à chaque enfant, mais chaque en-tête appartient à l'accordéon (et non à l'enfant). Lorsqu'un utilisateur clique sur un en-tête, l'enfant associé s'affiche sous cet en-tête. Le passage au nouvel enfant utilise une animation de transition.

Un accordéon doté d'enfants accepte le focus et modifie l'apparence de ses en-têtes afin d'afficher le focus. Lorsqu'un utilisateur emploie la tabulation dans un accordéon, l'en-tête sélectionné affiche l'indicateur de focus. Un accordéon sans enfants n'accepte pas le focus. Lorsque vous cliquez sur des composants qui acceptent le focus dans l'enfant sélectionné, ils reçoivent le focus. Lorsqu'une occurrence d'accordéon a le focus, vous pouvez utiliser les touches suivantes pour la contrôler :

Touche	Description
Flèche vers le bas, Flèche vers la droite	Place le focus sur l'en-tête de l'enfant suivant. Le focus passe du dernier en-tête au premier, sans modification de l'enfant sélectionné.
Flèche vers le haut, Flèche vers la gauche	Place le focus sur l'en-tête de l'enfant précédent. Le focus passe du premier en-tête au dernier, sans modification de l'enfant sélectionné.

Touche	Description
Fin	Sélectionne le dernier enfant.
Entrée/Espace	Sélectionne l'enfant associé à l'en-tête qui a le focus.
Origine	Sélectionne le premier enfant.
Pg. Suiv.	Sélectionne l'enfant suivant. La sélection passe du dernier enfant au premier.
Pg. Préc.	Sélectionne l'enfant précédent. La sélection passe du premier enfant au dernier.
Contrôle +	Place le focus sur le composant précédent. Ce composant peut être situé dans l'enfant sélectionné ou hors de l'accordéon, mais ne peut en aucun cas être un autre en-tête dans le même accordéon.
	Place le focus sur le composant suivant. Ce composant peut être situé dans l'enfant sélectionné ou hors de l'accordéon, mais ne peut en aucun cas être un autre en-tête dans le même accordéon.

Les lecteurs d'écran ne peuvent pas accéder au composant Accordion.

Utilisation du composant Accordion

Ce composant permet de présenter des formulaires comportant plusieurs parties. Par exemple, un accordéon de trois enfants peut présenter des formulaires que l'utilisateur remplit pour indiquer ses adresses de livraison et de facturation, ainsi que les données de paiement nécessaires à une transaction de commerce électronique. Le choix d'un accordéon à la place de plusieurs pages Web minimise le trafic sur le serveur et permet à l'utilisateur de mieux percevoir la progression et le contexte de la transaction.

Paramètres du composant Accordion

Dans l'inspecteur des propriétés ou l'inspecteur de composants (Fenêtre > Inspecteur de composants), vous pouvez définir les paramètres de création suivants pour chaque occurrence du composant Accordion :

childIcons est un tableau qui spécifie les identifiants de liaison des symboles de bibliothèque à utiliser comme icônes sur les en-têtes de l'accordéon. La valeur par défaut est `[]` (tableau vide).

childLabels est un tableau qui spécifie les étiquettes de texte à afficher sur les en-têtes de l'accordéon. La valeur par défaut est `[]` (tableau vide).

childNames est un tableau qui spécifie les noms d'occurrence des enfants de l'accordéon. Les valeurs saisies sont les noms des occurrences des symboles des enfants définis dans le paramètre childSymbols. La valeur par défaut est `[]` (tableau vide).

childSymbols est un tableau qui spécifie les identifiants de liaison des symboles de bibliothèque à utiliser pour créer les enfants de l'accordéon. La valeur par défaut est `[]` (tableau vide).

Dans l'inspecteur des composants, vous pouvez définir les paramètres supplémentaires suivants pour chaque occurrence du composant Accordion (Fenêtre > Inspecteur de composants) :

enabled est une valeur booléenne qui indique si le composant peut recevoir le focus et l'entrée. La valeur par défaut est `true`.

visible est une valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`). La valeur par défaut est `true`.

REMARQUE

Les propriétés minHeight et minWidth sont utilisées par des routines de dimensionnement internes. Elles sont définies dans UIObject et remplacées par différents composants selon vos besoins. Ces propriétés peuvent être utilisées pour créer un gestionnaire de dispositions personnalisées. Dans le cas contraire, la définition de ces propriétés dans l'inspecteur de composants n'a aucun effet visible.

Vous pouvez rédiger du code ActionScript pour contrôler d'autres options du composant Accordion en utilisant les propriétés, méthodes et événements ActionScript. Pour plus d'informations, voir « Classe Accordion », à la page 46.

Création d'une application avec le composant Accordion

Dans cet exemple, le développeur conçoit la section de paiement d'un magasin en ligne. L'application appelle un accordéon composé de trois formulaires, dans lesquels l'utilisateur entre son adresse de livraison, son adresse de facturation et ses données de paiement. Les formulaires d'adresse de livraison et d'adresse de facturation sont identiques.

Pour ajouter un composant Accordion à une application :

1. Sélectionnez Fichier > Nouveau, puis choisissez Fichier Flash (ActionScript 2.0).
2. Choisissez Insertion > Nouveau symbole et appelez-le **AddressForm**.
3. Si l'écran avancé n'est pas affiché, cliquez sur le bouton Avancé.
4. Sélectionnez Exporter pour ActionScript. Dans le champ Classe, entrez **mx.core.View**, puis cliquez sur OK.

Pour pouvoir conserver l'ordre de tabulation dans les enfants d'un accordéon, les enfants doivent également être des occurrences de la classe View.

5. Pour créer un formulaire d'adresse factice, faites glisser des composants, tels que Label et TextInput, du panneau Composants jusquà la Scène. Organisez-les et définissez leurs paramètres dans l'inspecteur des propriétés.

   Positionnez les éléments du formulaire sur la scène, par rapport aux coordonnées centrales (0,0). La coordonnée 0,0 du clip est placée dans le coin supérieur gauche de l'accordéon.

6. Choisissez Edition > Modifier le document pour revenir au scénario principal.

7. Répétez les étapes 2 à 6 pour créer un clip nommé **CheckoutForm**.

8. Faites glisser un composant Accordion à partir du panneau Composants pour l'ajouter à la scène dans le scénario principal.

9. Dans l'inspecteur des propriétés, procédez comme suit :

   - Entrez le nom d'occurrence **my_acc**.
   - Pour la propriété `childSymbols`, entrez **AddressForm**, **AddressForm** et **CheckoutForm**.

     Ces chaînes spécifient le nom des clips utilisés pour créer les enfants de l'accordéon.

REMARQUE	Les deux premiers enfants sont des occurrences du même clip car les formulaires d'adresse de livraison et d'adresse de facturation sont identiques.

   - Pour la propriété `childNames`, entrez **shippingAddress**, **billingAddress** et **checkout**.

     Ces chaînes sont les noms ActionScript des enfants de l'accordéon.

   - Pour la propriété `childLabels`, entrez **Adresse de livraison**, **Adresse de facturation** et **Paiement**.

     Ces chaînes sont les étiquettes de texte des en-têtes de l'accordéon.

   - Pour la propriété `childIcons`, entrez **IcôneAdresse**, **IcôneAdresse** et **IcônePaiement**.

     Ces chaînes spécifient les identifiants de liaison des symboles de clip qui sont utilisés comme icônes dans les en-têtes de l'accordéon. Vous devez créer ces symboles de clip si vous souhaitez que les en-têtes incluent des icônes.

10. Choisissez Contrôle > Tester l'animation.

Pour ajouter des enfants à un composant Accordion à l'aide d'ActionScript :

1. Sélectionnez Fichier > Nouveau, puis choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser un composant Accordion du panneau Composants vers la scène.
3. Dans l'inspecteur des propriétés, saisissez le nom d'occurrence **my_acc**.
4. Faites glisser un composant TextInput vers la bibliothèque.

   Cette opération ajoute le composant à la bibliothèque pour que vous puissiez l'instancier dynamiquement à l'step 6..
5. Dans le panneau Actions de l'image 1 du scénario, insérez le code suivant (il appelle la méthode createChild() pour créer ses affichages enfants) :

```
import mx.core.View;

// Création de panneaux enfants pour chaque formulaire à afficher
// dans l'objet my_acc.
my_acc.createChild(View, "shippingAddress", {label: "Shipping
  Address"});
my_acc.createChild(View, "billingAddress", {label: "Billing Address"});
my_acc.createChild(View, "payment", {label: "Payment"});
```

6. Dans le panneau Actions, sous le code que vous avez saisi à l'étape 5 dans l'image 1, entrez le code suivant (il ajoute deux occurrences de composant TextInput aux enfants du composant Accordion) :

```
// Création d'une saisie de texte enfant pour le panneau shippingAddress.
var firstNameChild_obj:Object =
  my_acc.shippingAddress.createChild("TextInput", "firstName", {text:
  "First Name"});

// Définition de la position de la saisie de texte.
firstNameChild_obj.move(10, 38);
firstNameChild_obj.setSize(110, 20);

// Création d'une autre saisie de texte enfant.
var lastNameChild_obj:Object =
  my_acc.shippingAddress.createChild("TextInput", "lastName", {text:
  "Last Name"});

// Définition de la position de la saisie de texte.
lastNameChild_obj.move(150, 38);
lastNameChild_obj.setSize(140, 20);
```

Personnalisation du composant Accordion

Vous pouvez transformer un composant Accordion horizontalement et verticalement au cours de la programmation et à l'exécution. Lors de la programmation, sélectionnez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez la méthode `setSize()` (voir `UIObject.setSize()`).

La méthode `setSize()` et l'outil de transformation modifient uniquement la largeur des en-têtes de l'accordéon, ainsi que la largeur et la hauteur de sa zone de contenu. La hauteur des en-têtes et la largeur et la hauteur des enfants n'en sont pas affectées. L'appel de la méthode `setSize()` est le seul moyen de modifier le cadre de délimitation d'un accordéon.

Si les en-têtes sont trop petits pour contenir le texte de leurs étiquettes, ces dernières sont rognées. Si la zone de contenu d'un accordéon est plus petite qu'un enfant, celui-ci est rogné.

Utilisation de styles avec le composant Accordion

Vous pouvez définir les propriétés des styles afin de modifier l'aspect de la bordure et de l'arrière-plan d'un composant Accordion.

Un composant Accordion utilise les styles suivants :

Style	Thème	Description
`themeColor`	Halo	Couleur de base d'un composant. Il s'agit du seul style de couleur qui n'hérite pas de sa valeur. "`haloGreen`", "`haloBlue`" et "`haloOrange`" sont des valeurs possibles.
`backgroundColor`	Les deux	Couleur d'arrière-plan. La couleur par défaut est le blanc.
`borderStyle`	Les deux	Le composant Accordion utilise une occurrence RectBorder en tant que bordure et répond au styles définis pour cette classe. Pour plus d'informations, voir « Classe RectBorder », à la page 1111. La valeur par défaut du style de bordure du composant Accordion est `solid`.
`headerHeight`	Les deux	Hauteur des boutons d'en-tête en pixels. La valeur par défaut est 22.
`color`	Les deux	Couleur du texte. La valeur par défaut est 0x0B333C pour le thème Halo et vide pour le thème Sample.
`disabledColor`	Les deux	Couleur du texte lorsque le composant est désactivé. La couleur par défaut est 0x848384 (gris foncé).

Style	Thème	Description
embedFonts	Les deux	Valeur booléenne qui indique si la police spécifiée dans fontFamily est de type intégré. Ce style doit être défini sur true si fontFamily fait référence à une police intégrée. Dans le cas contraire, la police intégrée n'est pas utilisée. Si ce style est défini sur true et que fontFamily ne fait pas référence à une police intégrée, aucun texte n'est affiché. La valeur par défaut est false.
fontFamily	Les deux	Nom de la police des étiquettes d'en-tête. La valeur par défaut est « _sans ».
fontSize	Les deux	Taille, en points, de la police des étiquettes d'en-tête. La valeur par défaut est 10.
fontStyle	Les deux	Style de police des étiquettes d'en-tête : « normal » ou « italic ». La valeur par défaut est « normal ».
fontWeight	Les deux	Epaisseur de la police des étiquettes d'en-tête : « none » ou « bold ». La valeur par défaut est « none ». Tous les composants peuvent également accepter la valeur « normal » au lieu de « none » pendant un appel à setStyle(), mais les prochains appels à getStyle() renvoient « none ».
textDecoration	Les deux	Décoration du texte : « none » ou « underline ».
openDuration	Les deux	Durée, en millisecondes, de l'animation de transition.
openEasing	Les deux	Référence à une fonction d'interpolation qui contrôle l'animation. Par défaut, sine in/out. Pour plus d'informations, reportez-vous à « Personnalisation des animations de composants » dans *Utilisation des composants ActionScript 2.0*.

Par conséquent, le code suivant définit en italique l'apparence de style de la police dans une occurrence Accordion appelée **my_acc** :

```
my_acc.setStyle("fontStyle", "italic");
```

Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, reportez-vous à « Utilisation de styles pour personnaliser la couleur et le texte des composants » dans *Utilisation des composants ActionScript 2.0*.

Utilisation d'enveloppes avec le composant Accordion

Le composant Accordion utilise des enveloppes pour représenter les différents états visuels de ses boutons d'en-tête. Pour envelopper les boutons et la barre de titre lors de la programmation, modifiez les symboles d'enveloppes dans le dossier Flash UI Components 2/Themes/MMDefault/Accordion Assets skins states, dans la bibliothèque de l'un des fichiers FLA de thèmes. Pour plus d'informations, reportez-vous à « A propos de l'application d'enveloppes aux composants » dans *Utilisation des composants ActionScript 2.0*.

Un composant Accordion est constitué de sa bordure, d'un arrière-plan, de boutons d'en-tête et de ses enfants. La bordure et l'arrière-plan sont fournis par la classe RectBorder par défaut. Pour plus d'informations sur l'application d'enveloppes à la classe RectBorder, reportez-vous à « Classe RectBorder », à la page 1111. Les en-têtes peuvent recevoir les enveloppes présentées ci-dessous.

Propriété	Description	Valeur par défaut
`falseUpSkin`	Etat relevé (normal) de l'en-tête, au-dessus des enfants réduits.	`accordionHeaderSkin`
`falseDownSkin`	Etat enfoncé de l'en-tête, au-dessus des enfants réduits.	`accordionHeaderSkin`
`falseOverSkin`	Etat survolé de l'en-tête, au-dessus des enfants réduits.	`accordionHeaderSkin`
`falseDisabled`	Etat désactivé de l'en-tête, au-dessus des enfants réduits.	`accordionHeaderSkin`
`trueUpSkin`	Etat relevé (normal) de l'en-tête, au-dessus de l'enfant développé.	`accordionHeaderSkin`
`trueDownSkin`	Etat enfoncé de l'en-tête, au-dessus de l'enfant développé.	`accordionHeaderSkin`
`trueOverSkin`	Etat survolé de l'en-tête, au-dessus de l'enfant développé.	`accordionHeaderSkin`
`trueDisabledSkin`	Etat désactivé de l'en-tête, au-dessus de l'enfant développé.	`accordionHeaderSkin`

Utilisation d'ActionScript pour le dessin de l'en-tête du composant Accordion

Les en-têtes par défaut des thèmes Halo et Sample utilisent le même élément d'enveloppe pour tous les états et tracent les vrais graphiques via ActionScript. L'implémentation de Halo utilise une extension de la classe RectBorder et un code d'API de dessin personnalisé pour tracer les états. L'implémentation de Sample utilise la même enveloppe et la même classe ActionScript que l'enveloppe du composant Button.

Pour créer une classe ActionScript, l'utiliser comme enveloppe et fournir différents états, l'enveloppe peut lire sa propriété de style `borderStyle` pour déterminer l'état. Le tableau suivant présente le style de bordure défini pour chaque enveloppe :

Propriété	Style de bordure
`falseUpSkin`	`falseup`
`falseDownSkin`	`falsedown`
`falseOverSkin`	`falserollover`
`falseDisabled`	`falsedisabled`
`trueUpSkin`	`trueup`
`trueDownSkin`	`truedown`
`trueOverSkin`	`truerollover`
`trueDisabledSkin`	`truedisabled`

Pour créer une enveloppe personnalisée par ActionScript pour l'en-tête d'accordéon :

1. Sélectionnez Fichier > Nouveau, puis choisissez Fichier ActionScript.
2. Copiez le code ActionScript suivant dans le fichier :

```
import mx.skins.RectBorder;
import mx.core.ext.UIObjectExtensions;

class RedGreenBlueHeader extends RectBorder
{
 static var symbolName_str:String = "RedGreenBlueHeader";
 static var symbolOwner_obj:Object = RedGreenBlueHeader;

 function size():Void
 {
  var color_num:Number; // Couleur
  var borderStyle_str:String = getStyle("borderStyle"); // Attribut
                                                        // d'Accordion
```

```
    // Définition des couleurs de chaque onglet dans l'Accordion
    // pour chaque état d'onglet.
    switch (borderStyle_str) {
     case "falseup":
     case "falserollover":
     case "falsedisabled":
      color_num = 0x7777FF;
      break;
     case "falsedown":
      color_num = 0x77FF77;
      break;
     case "trueup":
     case "truedown":
     case "truerollover":
     case "truedisabled":
      color_num = 0xFF7777;
      break;
    }

    // Effacement du style par défaut et tracé d'un style personnalisé.
    clear();
    lineStyle(0, 0, 100);
    beginFill(color_num, 100);
    drawRect(0, 0, __width, __height);
    endFill();
   }

   // obligatoire pour les enveloppes
   static function classConstruct():Boolean
   {
    UIObjectExtensions.Extensions();
    _global.skinRegistry["AccordionHeaderSkin"] = true;
    return true;
   }
   static var classConstructed_bl:Boolean = classConstruct();
   static var UIObjectExtensionsDependency_obj:Object =
    UIObjectExtensions;
  }
```

Cette classe crée une case carrée en fonction du style de bordure : une case bleue pour les états false relevé, survolé et désactivé, une case verte pour l'état enfoncé normal et une case rouge pour l'enfant développé.

3. Enregistrez le fichier.

   Dans cet exemple, nommez ce fichier **RedGreenBlueHeader.as**.

4. Sélectionnez Fichier > Nouveau, puis choisissez Fichier Flash (ActionScript 2.0).
5. Enregistrez le fichier dans le même dossier que le fichier AS.
6. Sélectionnez Insertion > Nouveau symbole et appelez-le **AccordionHeaderSkin**.

7. Si l'écran des paramètres avancés n'est pas affiché, cliquez sur le bouton Avancé.
8. Sélectionnez Exporter pour ActionScript.

   L'identifiant est automatiquement renseigné avec `AccordionHeaderSkin`.
9. Définissez la valeur de la classe sur `RedGreenBlueHeader`.
10. Assurez-vous que l'option Exporter dans la première image est bien activée, puis cliquez sur OK.
11. Dans Séquence 1, faites glisser un composant Accordion sur la scène.
12. Définissez les propriétés du composant Accordion pour qu'elles affichent plusieurs enfants.

    Par exemple, définissez `childLabels` sur un tableau `[One,Two,Three]` et `childNames` sur un tableau `[one,two,three]`.
13. Sélectionnez Contrôle > Tester l'animation.

Utilisation de clips pour personnaliser l'enveloppe d'en-tête du composant Accordion

L'exemple ci-dessus décrit l'utilisation d'une classe ActionScript 2.0 pour personnaliser l'enveloppe d'en-tête Accordion, méthode utilisée par les enveloppes des thèmes Halo et Sample. Toutefois, cet exemple utilise de simples cases colorées et il est plus judicieux dans ce cas d'employer des symboles de clip différents comme enveloppes d'en-tête.

Pour créer des symboles de clip pour les enveloppes d'en-tête du composant Accordion :

1. Sélectionnez Fichier > Nouveau, puis choisissez Fichier Flash (ActionScript 2.0).
2. Créez un symbole en choisissant Insertion > Nouveau symbole.
3. Nommez-le `RedAccordionHeaderSkin`.
4. Si l'écran des paramètres avancés n'est pas affiché, cliquez sur le bouton Avancé.
5. Sélectionnez Exporter pour ActionScript.

   L' identifiant est automatiquement renseigné avec `RedAccordionHeaderSkin`.
6. Laissez le champ de texte Classe vide.
7. Assurez-vous que l'option Exporter dans la première image est bien activée, puis cliquez sur OK.
8. Ouvrez le nouveau symbole pour l'édition.
9. Servez-vous des outils de dessin pour créer une case rouge entourée d'une ligne noire.
10. Définissez le style de bordure sur Filet.

11. Définissez la case, et sa bordure, de sorte qu'elle soit positionnée au point (0,0) avec une largeur et une hauteur de 100.

    Le code ActionScript dimensionne l'enveloppe de la manière appropriée.

12. Répétez les étapes 2 à 11 et créez des enveloppes verte et bleue, nommées en conséquence.

13. Cliquez sur le bouton Précédent pour revenir au scénario principal.

14. Faites glisser un composant Accordion sur la scène.

15. Définissez les propriétés du composant Accordion pour qu'elles affichent plusieurs enfants.

    Par exemple, définissez `childLabels` sur un tableau `[One,Two,Three]` et `childNames` sur un tableau `[one,two,three]`.

16. L'occurrence Accordion étant sélectionnée, copiez le code ActionScript suivant dans le panneau Actions :

```
onClipEvent(initialize) {
  falseUpSkin = "RedAccordionHeaderSkin";
  falseDownSkin = "GreenAccordionHeaderSkin";
  falseOverSkin = "RedAccordionHeaderSkin";
  falseDisabled = "RedAccordionHeaderSkin";
  trueUpSkin = "BlueAccordionHeaderSkin";
  trueDownSkin = "BlueAccordionHeaderSkin";
  trueOverSkin = "BlueAccordionHeaderSkin";
  trueDisabledSkin = "BlueAccordionHeaderSkin";
}
```

17. Choisissez Contrôle > Tester l'animation.

Classe Accordion

Héritage MovieClip > Classe UIObject > Classe UIComponent > Affichage > Accordion

Nom de classe ActionScript mx.containers.Accordion

Un composant Accordion contient des enfants qui s'affichent l'un après l'autre. A chaque enfant est associé un bouton d'en-tête correspondant, créé en même temps que l'enfant. L'enfant doit être une occurrence de la classe UIObject.

REMARQUE	Le composant Accordion est pris en charge uniquement si vous travaillez dans un document ActionScript 2.0.

Un symbole de clip devient automatiquement une occurrence de la classe UIObject lorsqu'il devient l'enfant d'un accordéon. Cependant, pour conserver l'ordre de tabulation dans les enfants d'un accordéon, les enfants doivent être aussi des occurrences de la classe View. Si vous utilisez un symbole de clip comme enfant, définissez son champ Classe sur mx.core.View pour qu'il hérite de la classe View.

La définition d'une propriété de la classe Accordion avec ActionScript annule le paramètre du même nom défini dans l'inspecteur des propriétés ou des composants.

Toutes les classes de composants ont une propriété `version` qui correspond à une propriété de classe. Les propriétés de classe ne sont disponibles que pour la classe elle-même. La propriété `version` renvoie une chaîne qui indique la version du composant. Pour accéder à cette propriété, utilisez le code suivant :

```
trace(mx.containers.Accordion.version);
```

REMARQUE

Le code `trace(my_accInstance.version);` renvoie `undefined`.

Méthodes de la classe Accordion

Le tableau suivant présente les méthodes de la classe Accordion.

Méthode	Description
`Accordion.createChild()`	Crée un enfant pour une occurrence Accordion.
`Accordion.createSegment()`	Crée un enfant pour une occurrence Accordion. Ses paramètres sont différents de ceux de la méthode `createChild()`.
`Accordion.destroyChildAt()`	Détruit un enfant à la position d'index spécifiée.
`Accordion.getChildAt()`	Obtient une référence à un enfant à une position d'index spécifiée.
`Accordion.getHeaderAt()`	Obtient une référence à un objet d'en-tête à une position d'index spécifiée.

Méthodes héritées de la classe UIObject

Le tableau suivant énumère les méthodes de la classe Accordion héritées de la classe UIObject. Lors de l'appel à ces méthodes à partir de l'objet Accordion, utilisez le formulaire *accordionInstance.methodName*.

Méthode	Description
UIObject.createClassObject()	Crée un objet dans la classe spécifiée.
UIObject.createObject()	Crée un sous-objet dans un objet.
UIObject.destroyObject()	Détruit une occurrence de composant.
UIObject.doLater()	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
UIObject.getStyle()	Obtient la propriété de style de l'objet ou de la déclaration de style.
UIObject.invalidate()	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
UIObject.move()	Déplace l'objet jusqu'à l'emplacement demandé.
UIObject.redraw()	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
UIObject.setSize()	Redimensionne l'objet à la taille demandée.
UIObject.setSkin()	Définit une enveloppe dans l'objet.
UIObject.setStyle()	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant énumère les méthodes de la classe Accordion héritées de la classe UIComponent. Lors de l'appel à ces méthodes à partir de l'objet Accordion, utilisez le formulaire *accordionInstance.methodName*.

Méthode	Description
UIComponent.getFocus()	Renvoie une référence à l'objet ayant le focus.
UIComponent.setFocus()	Définit le focus sur l'occurrence de composant.

Propriétés de la classe Accordion

Le tableau suivant présente les propriétés de la classe Accordion.

Propriété	Description
`Accordion.numChildren`	Nombre d'enfants d'une occurrence Accordion.
`Accordion.selectedChild`	Référence à l'enfant sélectionné.
`Accordion.selectedIndex`	Position d'index de l'enfant sélectionné.

Propriétés héritées de la classe UIObject

Le tableau suivant énumère les propriétés de la classe Accordion héritées de la classe UIObject. Lorsque vous accédez à ces propriétés, utilisez le formulaire *`accordionInstance.propertyName`*.

Propriété	Description
`UIObject.bottom`	Lecture seule ; position du bord inférieur de l'objet par rapport au bord inférieur de son parent.
`UIObject.height`	Lecture seule ; hauteur de l'objet, en pixels.
`UIObject.left`	Lecture seule ; bord gauche de l'objet, en pixels.
`UIObject.right`	Lecture seule ; position du bord droit de l'objet par rapport au bord droit de son parent.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Lecture seule ; renvoie la position du bord supérieur de l'objet par rapport à son parent.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Lecture seule ; largeur de l'objet, en pixels.
`UIObject.x`	Lecture seule ; bord gauche de l'objet, en pixels.
`UIObject.y`	Lecture seule ; bord supérieur de l'objet, en pixels.

Propriétés héritées de la classe UIComponent

Le tableau suivant énumère les propriétés de la classe Accordion héritées de la classe UIComponent. Lorsque vous accédez à ces propriétés, utilisez le formulaire *accordionInstance.propertyName*.

Propriété	Description
`UIComponent.enabled`	Indique si le composant peut recevoir le focus et la saisie.
`UIComponent.tabIndex`	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Evénements de la classe Accordion

Le tableau suivant présente un événement de la classe Accordion.

Evénement	Description
`Accordion.change`	Diffusé à tous les écouteurs enregistrés lorsque les propriétés `selectedIndex` et `selectedChild` d'un accordéon changent parce que l'utilisateur a cliqué sur la souris ou appuyé sur une touche.

Evénements hérités de la classe UIObject

Le tableau suivant énumère les événements de la classe Accordion hérités de la classe UIObject.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant énumère les événements de la classe Accordion hérités de la classe UIComponent.

Evénement	Description
`UIComponent.focusIn`	Diffusé lorsqu'un objet reçoit le focus.
`UIComponent.focusOut`	Diffusé lorsqu'un objet perd le focus.
`UIComponent.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIComponent.keyUp`	Diffusé lorsqu'une touche est relâchée.

Accordion.change

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.change = function(eventObject:Object) {
  // Insérer votre code ici.
};
accordionInstance.addEventListener("change", listenerObject);
```

Utilisation 2 :

```
on (change) {
  // Insérer votre code ici.
}
```

Description

Evénement : diffusé à l'ensemble des écouteurs enregistrés lorsque les propriétés `selectedIndex` et `selectedChild` d'un accordéon changent. Cet événement est diffusé uniquement lorsqu'un clic de souris ou une pression sur une touche modifie la valeur `selectedChild` ou `selectedIndex` (et non lorsque la valeur est modifiée par ActionScript). Cet événement est diffusé avant l'animation de transition.

Les composants utilisent un modèle dispatcher(diffuseur)/écouteur d'événement. Le composant Accordion distribue un événement `change` en cas de clic sur l'un de ses boutons, et l'événement est géré par une fonction (également appelée *gestionnaire*) sur un objet écouteur (*`listenerObject`*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez une référence au gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`eventObject`*) au gestionnaire. Chaque objet événement possède des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés dans le code qui traitera l'événement. Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

L'événement change du composant Accordion contient également deux propriétés d'objet événement uniques :

- `newValue` Nombre correspondant à l'index de l'enfant qui va être sélectionné.
- `prevValue` Nombre correspondant à l'index de l'enfant sélectionné précédemment.

Exemple

L'exemple suivant utilise une occurrence Accordion appelée `my_acc` qui contient trois panneaux enfants intitulés « Adresse de livraison », « Adresse de facturation » et « Paiement ». Le code définit un gestionnaire appelé `my_accListener` et le transmet à la méthode `my_acc.addEventListener()` comme deuxième paramètre. L'objet événement est capturé par le gestionnaire `change` du paramètre *`eventObject`*. Lorsque l'événement `change` est diffusé, une instruction `trace` est envoyée au panneau Sortie.

```
// Création d'un objet écouteur.
var my_accListener:Object = new Object();
my_accListener.change = function() {
  trace("Changed to different view");
  // Affectation de l'étiquette du panneau enfant à la variable.
  var selectedChild_str:String = my_acc.selectedChild.label;
  // Exécution de l'action basée sur l'enfant sélectionné.
  switch (selectedChild_str) {
  case "Shipping Address":
    trace("One was selected");
    break;
  case "Billing Address":
    trace("Two was selected");
    break;
  case "Payment":
    trace("Three was selected");
    break;
  }
};
my_acc.addEventListener("change", my_accListener);
```

Accordion.createChild()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
accordionInstance.createChild(classOrSymbolName, instanceName[,
  initialProperties])
```

Paramètres

classOrSymbolName Soit la fonction constructeur de la classe de UIObject à instancier, soit le nom de liaison (référence au symbole à instancier). La classe doit être UIObject ou une sous-classe de UIObject, mais le plus souvent, il s'agit d'un objet View ou d'une sous-classe de View.

instanceName Nom de la nouvelle occurrence.

initialProperties Paramètre facultatif qui indique les propriétés initiales de la nouvelle occurrence. Vous pouvez utiliser les propriétés suivantes :

- `label` Chaîne qui spécifie l'étiquette de texte que la nouvelle occurrence de l'enfant affiche sur son en-tête.
- `icon` Chaîne qui spécifie l'identifiant de liaison du symbole de bibliothèque que l'enfant utilise pour l'icône affichée sur son en-tête.

Valeur renvoyée

Une référence à une occurrence de UIObject qui est l'enfant venant d'être créé.

Description

Méthode (héritée de View) qui crée un enfant pour l'accordéon. L'enfant nouvellement créé est ajouté à la fin de la liste des enfants que possède l'accordéon. Utilisez cette méthode pour placer des affichages dans l'accordéon. L'enfant créé est une occurrence du symbole de clip ou de classe spécifié dans le paramètre *classOrSymbolName*. Pour chaque enfant, vous pouvez utiliser les propriétés `label` et `icon` pour désigner une étiquette de texte et une icône destinées à l'en-tête d'accordéon associé dans le paramètre *initialProperties*.

Lors de sa création, chaque enfant se voit attribuer un numéro d'index dans l'ordre de création, et la propriété `numChildren` est incrémentée de 1.

Exemple

Commencez avec une occurrence Accordion sur la scène appelée `my_acc`. Ajoutez un symbole à la bibliothèque avec l'identifiant de liaison `payIcon` comme icône de l'en-tête enfant. Le code suivant crée un enfant appelé `billing` (avec l'étiquette « Payment ») qui est une occurrence de la classe View :

```
var child_obj:Object = my_acc.createChild(mx.core.View, "billing", {label:
  "Payment", icon: "payIcon"});
```

Le code suivant crée également un enfant qui est une occurrence de la classe View, mais il utilise `import` pour référencer le constructeur de la classe View :

```
import mx.core.View;
var child_obj:Object = my_acc.createChild(View, "billing", {label:
  "Payment", icon: "payIcon"});
```

Vous pouvez également ajouter un symbole de clip à la bibliothèque avec l'identifiant de liaison `PaymentForm` comme enfant du composant Accordion, et le code suivant crée une occurrence de `PaymentForm` appelée `billing` comme enfant de `my_acc` (cette approche est utile lorsque vous chargez le contenu dynamique dans un symbole de clip, puis faites de ce symbole un enfant de l'occurrence Accordion) :

```
var child_obj:Object = my_acc.createChild("PaymentForm", "billing", {label:
  "Payment", icon: "payIcon"});
```

Pour un exemple plus complexe, conservez l'occurrence Accordion `my_acc` sur la scène. Faites ensuite glisser un composant Label dans un composant TextInput depuis le panneau Composants vers la bibliothèque du document actuel (de façon à ce que la bibliothèque contienne à la fois un symbole TextInput et un symbole Label). Collez le code suivant dans la première image du scénario principal (en remplaçant ainsi le code des exemples précédents). Le code suivant crée un enfant qui est une occurrence de la classe View appelée `billing` et ajoute également des enfants à `billing` pour fournir des étiquettes et des champs de saisie de texte pour un formulaire :

```
import mx.core.View;
import mx.controls.Label;
import mx.controls.TextInput;
var child_obj:Object = my_acc.createChild(View, "billing",
  {label:"Payment", icon: "payIcon"});
// Création d'étiquettes comme enfants de l'occurrence d'affichage.
var cardType_label:Object = child_obj.createChild(Label, "CardType_label",
  {_x:10, _y:50});
var cardNumber_label:Object = child_obj.createChild(Label,
  "CardNumber_label", {_x:10, _y:100});
// Création de saisies de texte comme enfants de l'occurrence d'affichage.
var cardTypeInput_ti:Object = child_obj.createChild(TextInput,
  "CardType_ti", {_x:150, _y:50});
```

```
var cardNumberInput_ti:Object = child_obj.createChild(TextInput,
  "CardNumber_ti", {_x:150, _y:100});
// Remplissage des étiquettes.
cardType_label.text = "Card Type";
cardNumber_label.text = "Card Number";
```

Accordion.createSegment()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
accordionInstance.createSegment(classOrSymbolName, instanceName[, label[,
  icon]])
```

Paramètres

classOrSymbolName Soit une référence à la fonction constructeur de la classe de UIObject à instancier, soit le nom de liaison du symbole à instancier. La classe doit être UIObject ou une sous-classe de UIObject, mais le plus souvent, il s'agit de View ou d'une sous-classe de View.

instanceName Nom de la nouvelle occurrence.

label Cette chaîne spécifie l'étiquette de texte que la nouvelle occurrence de l'enfant affiche sur son en-tête. Ce paramètre est facultatif.

icon Cette chaîne fait référence à l'identifiant de liaison du symbole de bibliothèque que l'enfant utilise pour l'icône affichée sur son en-tête. Ce paramètre est facultatif.

Valeur renvoyée

Une référence à la nouvelle occurrence UIObject.

Description

Méthode qui crée un enfant pour l'accordéon. L'enfant nouvellement créé est ajouté à la fin de la liste des enfants que possède l'accordéon. Utilisez cette méthode pour placer des affichages dans l'accordéon. L'enfant créé est une occurrence du symbole de clip ou de classe spécifié dans le paramètre *classOrSymbolName*. Vous pouvez utiliser les paramètres *label* et *icon* pour spécifier une étiquette de texte ainsi qu'une icône pour l'en-tête de l'accordéon associé à chaque enfant.

La méthode `createSegment()` diffère de la méthode `createChild()` dans la mesure où *`label`* et *`icon`* sont transmis directement en tant que paramètres, et non en tant que propriétés d'un paramètre *`initalProperties`*.

Lors de sa création, chaque enfant se voit attribuer un numéro d'index dans l'ordre de création, et la propriété `numChildren` est incrémentée de 1.

Exemple

Commencez avec une occurrence Accordion sur la scène appelée `my_acc`. Ajoutez un symbole de clip à la bibliothèque avec l'identifiant de liaison `PaymentForm` comme enfant du composant Accordion. Ajoutez ensuite un symbole à la bibliothèque avec l'identifiant de liaison `payIcon` comme icône de l'en-tête enfant. L'exemple suivant crée une occurrence du symbole de clip `PaymentForm` appelée `billing` comme dernier enfant de `my_acc` avec l'étiquette d'en-tête « Payment » dans la bibliothèque :

```
var child_obj:Object = my_acc.createSegment("PaymentForm", "billing",
  "Payment", "payIcon");
```

Le code suivant crée un enfant qui est une occurrence de la classe View :

```
var child_obj:Object = my_acc.createSegment(mx.core.View, "billing",
  "Payment", "payIcon");
```

Le code suivant crée également un enfant qui est une occurrence de la classe View, mais il utilise `import` pour référencer le constructeur de la classe View :

```
import mx.core.View;
var child_obj:Object = my_acc.createSegment(View, "billing", "Payment",
  "payIcon");
```

Faites glisser un composant Label et un composant TextInput du panneau Composants vers la bibliothèque du document en cours (afin que la bibliothèque contienne à la fois un symbole TextInput et un symbole Label). Le code suivant crée un enfant qui est une occurrence de la classe View appelée `billing` et ajoute également des enfants à `billing` pour fournir des étiquettes et des champs de saisie de texte pour un formulaire :

```
import mx.core.View;
import mx.controls.Label;
import mx.controls.TextInput;
var child_obj:Object = my_acc.createSegment(View, "billing", "Payment",
  "payIcon");
// Création d'étiquettes comme enfants de l'occurrence d'affichage.
var cardType_label:Object = child_obj.createChild(Label, "CardType_label",
  {_x:10, _y:50});
var cardNumber_label:Object = child_obj.createChild(Label,
  "CardNumber_label", {_x:10, _y:100});
// Création de saisies de texte comme enfants de l'occurrence d'affichage.
var cardTypeInput_ti:Object = child_obj.createChild(TextInput,
  "CardType_ti", {_x:150, _y:50});
```

```
var cardNumberInput_ti:Object = child_obj.createChild(TextInput,
  "CardNumber_ti", {_x:150, _y:100});
// Remplissage des étiquettes.
cardType_label.text = "Card Type";
cardNumber_label.text = "Card Number";
```

Accordion.destroyChildAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`accordionInstance.destroyChildAt(`*`index`*`)`

Paramètres

`index` Numéro d'index de l'enfant de l'accordéon à détruire. Un numéro d'index basé sur zéro est affecté à chaque enfant d'un accordéon, en fonction de l'ordre de sa création.

Valeur renvoyée

Aucune.

Description

Méthode (héritée de View) ; détruit l'un des enfants de l'accordéon. L'enfant à détruire est spécifié par son index, lequel est transmis à la méthode dans le paramètre *`index`*. L'appel de cette méthode détruit également l'en-tête correspondant.

Si l'enfant détruit est sélectionné, un nouvel enfant sélectionné est choisi. S'il existe un enfant suivant, il est sélectionné. S'il n'existe aucun enfant suivant, l'enfant précédent est sélectionné. S'il n'existe aucun enfant précédent, la sélection est « undefined ».

REMARQUE	L'appel à la méthode `destroyChildAt()` décrémente la propriété `numChildren` de 1.

Exemple

Le code suivant détruit le premier enfant de `my_acc` lorsque le troisième enfant est sélectionné :

```
import mx.core.View;

// Création de panneaux enfants avec des occurrences de la classe View.
my_acc.createSegment(View, "myMainItem1", "Menu Item 1");
my_acc.createSegment(View, "myMainItem2", "Menu Item 2");
my_acc.createSegment(View, "myMainItem3", "Menu Item 3");

// Création d'un objet écouteur.
my_accListener = new Object();
my_accListener.change = function() {
   if ("myMainItem3"){
   my_acc.destroyChildAt(0);
   }
};

my_acc.addEventListener("change", my_accListener);
```

Voir aussi

`Accordion.createChild()`

Accordion.getChildAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`accordionInstance.getChildAt(`*`index`*`)`

Paramètres

`index` Numéro d'index d'un enfant d'accordéon. Un numéro d'index basé sur zéro est affecté à chaque enfant d'un accordéon, en fonction de l'ordre de sa création.

Valeur renvoyée

Référence à l'occurrence UIObject à l'emplacement d'index spécifié.

Description

Méthode qui renvoie une référence à l'enfant à l'emplacement d'index spécifié. Un numéro d'index est attribué à chaque enfant de l'accordéon pour son emplacement. Ce numéro d'index est basé sur zéro : le premier enfant est 0, le second enfant est 1, et ainsi de suite.

Exemple

Le code suivant obtient une référence au dernier enfant de `my_acc` et change l'étiquette pour « Last Child » :

```
import mx.core.View;

// Création de panneaux enfants avec des occurrences de la classe View.
my_acc.createSegment(View, "myMainItem1", "Menu Item 1");
my_acc.createSegment(View, "myMainItem2", "Menu Item 2");
my_acc.createSegment(View, "myMainItem3", "Menu Item 3");

// Obtention de la référence pour le dernier objet enfant.
var lastChild_obj:Object = my_acc.getChildAt(my_acc.numChildren - 1);
// Changement de l'étiquette de l'objet.
lastChild_obj.label = "Last Child";
```

Accordion.getHeaderAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`accordionInstance.getHeaderAt(`*`index`*`)`

Paramètres

`index` Numéro d'index d'un en-tête d'accordéon. Un numéro d'index basé sur zéro est affecté à chaque en-tête d'un accordéon, en fonction de l'ordre de sa création.

Valeur renvoyée

Référence à l'occurrence UIObject à l'emplacement d'index spécifié.

Description

Méthode ; renvoie une référence à l'en-tête à l'emplacement d'index spécifié. Un numéro d'index est attribué à chaque en-tête de l'accordéon pour son emplacement. Ce numéro d'index est basé sur zéro : le premier en-tête est 0, le deuxième en-tête est 1, et ainsi de suite.

Exemple

Le code suivant obtient une référence au dernier en-tête de `my_acc` et affiche l'étiquette dans le panneau Sortie :

```
import mx.core.View;

// Création de panneaux enfants pour chaque formulaire à afficher
// dans l'objet my_acc.
my_acc.createChild(View, "shippingAddress", {label: "Shipping Address"});
my_acc.createChild(View, "billingAddress", {label: "Billing Address"});
my_acc.createChild(View, "payment", {label: "Payment"});

var head3:Object = my_acc.getHeaderAt(2);
trace(head3.label);
```

Accordion.numChildren

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
accordionInstance.numChildren
```

Description

La propriété (héritée de View) indique le nombre d'enfants (de type UIObject) présents dans une occurrence Accordion. Les en-têtes ne sont pas comptés comme enfants.

Un numéro d'index est attribué à chaque enfant de l'accordéon pour son emplacement. Ce numéro d'index est basé sur zéro : le premier enfant est 0, le second enfant est 1, et ainsi de suite. Le code `my_acc.numChild - 1` fait toujours référence au dernier enfant ajouté à un accordéon. Par exemple, s'il existe sept enfants dans un accordéon, le dernier a l'index 6. La propriété `numChildren` n'étant pas basée sur zéro, la valeur de `my_acc.numChildren` est 7. Le résultat de 7 - 1 est 6, le numéro d'index du dernier enfant.

Exemple

Le code suivant utilise `numChildren` pour obtenir une référence au dernier enfant de `my_acc` et change l'étiquette sur « Last Child » :

```
import mx.core.View;

// Création de panneaux enfants avec des occurrences de la classe View.
my_acc.createSegment(View, "myMainItem1", "Menu Item 1");
my_acc.createSegment(View, "myMainItem2", "Menu Item 2");
my_acc.createSegment(View, "myMainItem3", "Menu Item 3");

// Obtention de la référence pour le dernier objet enfant.
var lastChild_obj:Object = my_acc.getChildAt(my_acc.numChildren - 1);
// Changement de l'étiquette de l'objet.
lastChild_obj.label = "Last Child";
```

Accordion.selectedChild

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
accordionInstance.selectedChild
```

Description

Propriété : enfant sélectionné (de type UIObject) s'il existe un ou plusieurs enfants ; `undefined` s'il n'en existe aucun.

Si l'accordéon a des enfants, le code `accordionInstance.selectedChild` est équivalent au code `accordionInstance.getChildAt(accordionInstance.selectedIndex)`.

Lorsque vous définissez cette propriété pour un enfant, l'accordéon commence l'animation de transition pour afficher l'enfant désigné.

La modification de la valeur de `selectedChild` change également la valeur de `selectedIndex`.

Si l'accordéon a des enfants, la valeur par défaut est `accordionInstance.getChildAt(0)`. S'il n'en a pas, la valeur par défaut est `undefined`.

Exemple

L'exemple suivant détecte lorsqu'un enfant est sélectionné et affiche l'ordre de l'enfant dans le panneau Sortie chaque fois qu'un en-tête est sélectionné :

```
// Création d'un objet écouteur.
var my_accListener:Object = new Object();
my_accListener.change = function() {
  trace("Changed to different view");
  // Affectation de l'étiquette du panneau enfant à la variable.
  var selectedChild_str:String = my_acc.selectedChild.label;
  // Exécution de l'action basée sur l'enfant sélectionné.
  switch (selectedChild_str) {
  case "Shipping Address":
    trace("One was selected");
    break;
  case "Billing Address":
    trace("Two was selected");
    break;
  case "Payment":
    trace("Three was selected");
    break;
  }
};
my_acc.addEventListener("change", my_accListener);
```

Voir aussi

`Accordion.selectedIndex`

Accordion.selectedIndex

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
accordionInstance.selectedIndex
```

Description

Propriété : index basé sur zéro de l'enfant sélectionné dans un accordéon comportant un ou plusieurs enfants. Dans le cas d'un accordéon n'ayant aucun affichage enfant, la seule valeur valide est `undefined`.

Un numéro d'index est attribué à chaque enfant de l'accordéon pour son emplacement. Ce numéro d'index est basé sur zéro : le premier enfant est 0, le second enfant est 1, et ainsi de suite. Les valeurs valides de `selectedIndex` sont 0, 1, 2, etc., *n* - 1, où *n* correspond au nombre d'enfants.

Lorsque vous définissez cette propriété sur un enfant, l'accordéon commence l'animation de transition pour afficher l'enfant désigné.

La modification de la valeur de `selectedIndex` change également la valeur de `selectedChild`.

Exemple

L'exemple suivant détecte lorsqu'un enfant est sélectionné et affiche l'ordre attribué à l'enfant dans le panneau Sortie chaque fois qu'un en-tête est sélectionné :

```
// Création d'un objet Listener (écouteur).
var my_accListener:Object = new Object();
my_accListener.change = function() {
  trace("Changed to different view");
  // Affectation de l'étiquette du panneau enfant à la variable.
  var selectedChild_num:Number = my_acc.selectedIndex;
  // Exécution de l'action basée sur l'enfant sélectionné.
  switch (selectedChild_num) {
  case 0:
    trace("One was selected");
    break;
  case 1:
    trace("Two was selected");
    break;
  case 2:
    trace("Three was selected");
    break;
  }
};
my_acc.addEventListener("change", my_accListener);
```

Voir aussi

`Accordion.numChildren`, `Accordion.selectedChild`

CHAPITRE 3

Composant Alert

3

Le composant Alert vous permet d'ouvrir une fenêtre qui présente à l'utilisateur un message et des boutons de réponse. Cette fenêtre comprend une barre de titre dans laquelle vous pouvez insérer du texte, un message personnalisable et des boutons dont les étiquettes peuvent être modifiées. Une fenêtre Alert peut combiner librement les boutons Oui, Non, OK et Annuler, et les propriétés `Alert.okLabel`, `Alert.yesLabel`, `Alert.noLabel` et `Alert.cancelLabel` vous permettent de modifier leurs étiquettes. Dans une fenêtre Alert, vous ne pouvez cependant pas modifier l'ordre des boutons qui est toujours OK, Oui, Non, Annuler. Dès que l'utilisateur clique sur l'un de ces boutons, la fenêtre Alert se ferme.

REMARQUE

Le composant Alert est pris en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Pour afficher une fenêtre Alert, appelez la méthode `Alert.show()`. Cet appel ne fonctionne que si le composant Alert figure déjà dans la bibliothèque. Pour ajouter le composant Alert à la bibliothèque sans qu'il soit visible dans le document, faites-le glisser du panneau Composants vers la scène, puis supprimez-le.

L'aperçu en direct du composant Alert est une fenêtre vide.

Lorsque vous ajoutez le composant Alert à une application, utilisez le panneau Accessibilité pour rendre son texte et ses boutons accessibles aux logiciels de lecture d'écran. Commencez par ajouter la ligne de code suivante pour activer l'accessibilité :

```
mx.accessibility.AlertAccImpl.enableAccessibility();
```

CONSEIL

Quel que soit le nombre d'occurrences du composant, l'activation de son accessibilité ne se fait qu'une fois.

Utilisation du composant Alert

Vous pouvez utiliser un composant Alert chaque fois que vous voulez annoncer quelque chose à l'utilisateur. Par exemple, vous pouvez faire apparaître une alerte lorsqu'un utilisateur ne remplit pas un formulaire correctement, lorsqu'une action atteint une certaine valeur ou lorsqu'un utilisateur quitte une application sans enregistrer sa session.

Paramètres du composant Alert

Le composant Alert n'a pas de paramètres de programmation. Pour afficher une fenêtre Alert, vous devez appeler la méthode `Alert.show()` d'ActionScript. Vous pouvez utiliser d'autres propriétés ActionScript pour modifier la fenêtre Alert dans une application. Pour plus d'informations, voir « Classe Alert », à la page 71.

Création d'une application avec le composant Alert

La procédure suivante décrit l'ajout d'un composant Alert à une application lors de la programmation. Dans cet exemple, le composant Alert apparaît lorsqu'une action atteint une certaine valeur sur le marché boursier.

Pour créer une application avec le composant Alert :

1. Sélectionnez Fichier > Nouveau, puis choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser le composant Alert du panneau Composants jusqu'à la bibliothèque du document actuel.

   Cette opération permet d'ajouter le composant à la bibliothèque, mais elle ne permet pas de le rendre visible dans l'application.
3. Dans le panneau Actions, entrez le code suivant dans l'image 1 du scénario pour définir un gestionnaire d'événements pour l'événement `click` :

```
import mx.controls.Alert;

// Définition de l'action après la confirmation de l'alerte.
var myClickHandler:Function = function (evt_obj:Object) {
 if (evt_obj.detail == Alert.OK) {
  trace("start stock app");
 }
};

// Affichage de la boîte de dialogue Alerte.
Alert.show("Launch Stock Application?", "Stock Price Alert", Alert.OK |
  Alert.CANCEL, this, myClickHandler, "stockIcon", Alert.OK);
```

Ce code crée une fenêtre Alert incluant les boutons OK et Annuler. Lorsque l'utilisateur clique sur l'un des boutons, Flash appelle la fonction `myClickHandler`. Cette fonction indique à Flash de tracer « start stock app » lorsque vous cliquez sur le bouton OK.

REMARQUE	La méthode `Alert.show()` inclut un paramètre facultatif qui affiche une icône dans la fenêtre Alert (dans cet exemple, une icône avec l'identifiant de liaison « stockIcon »). Pour inclure cette icône dans votre exemple de test, créez un symbole nommé stockIcon et activez l'option Exporter pour ActionScript de la boîte de dialogue Propriétés de liaison ou Créer un symbole. Les graphiques du symbole stockIcon doivent être alignés sur les coordonnées (0,0) dans le système de coordonnées du symbole.

4. Choisissez Contrôle > Tester l'animation.

Personnalisation du composant Alert

Le composant Alert se positionne automatiquement au centre du composant transmis comme paramètre *`parent`*. Le parent doit être un objet UIComponent. S'il s'agit d'un clip, vous pouvez l'enregistrer sous mx.core.View afin qu'il hérite de UIComponent.

La fenêtre Alert s'étend horizontalement pour pouvoir contenir le texte du message ou les boutons qui s'affichent. Si vous voulez afficher une grande quantité de texte, incluez des sauts de ligne.

Le composant Alert ne répond pas à la méthode `setSize()`.

Utilisation de styles avec le composant Alert

Vous pouvez définir des propriétés de style pour modifier l'aspect d'un composant Alert. Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, consultez « Utilisation de styles pour personnaliser la couleur et le texte des composants » dans *Utilisation des composants ActionScript 2.0*.

Un composant Alert gère les styles suivants :

Style	Thème	Description
`themeColor`	Halo	Couleur de base d'un composant. `"haloGreen"`, `"haloBlue"` et `"haloOrange"` sont des valeurs possibles. La valeur par défaut est `"haloGreen"`.
`backgroundColor`	Les deux	Couleur d'arrière-plan. Le blanc est la couleur par défaut pour le thème Halo et 0xEFEBEF (gris clair) pour le thème Sample.

Style	Thème	Description
`borderStyle`	Les deux	Le composant Alert utilise une occurrence RectBorder en tant que bordure et répond aux styles définis pour cette classe. Pour plus d'informations, voir « Classe RectBorder », à la page 1111. Le composant Alert présente un paramètre `borderStyle` propre au composant, `"alert"` avec le thème Halo et `"outset"` avec le thème Sample.
`color`	Les deux	Couleur du texte. La valeur par défaut est 0x0B333C pour le thème Halo et vide pour le thème Sample.
`disabledColor`	Les deux	Couleur du texte lorsque le composant est désactivé. La couleur par défaut est 0x848384 (gris foncé).
`embedFonts`	Les deux	Valeur booléenne qui indique si la police spécifiée dans `fontFamily` est de type intégré. Ce style doit être défini sur `true` si `fontFamily` fait référence à une police intégrée. Dans le cas contraire, la police intégrée n'est pas utilisée. Si ce style est défini sur `true` et que `fontFamily` ne fait pas référence à une police intégrée, aucun texte n'est affiché. La valeur par défaut est `false`.
`fontFamily`	Les deux	Nom de la police du texte. La valeur par défaut est `"_sans"`.
`fontSize`	Les deux	Taille, en points, de la police. La valeur par défaut est 10.
`fontStyle`	Les deux	Style de police : `"normal"` ou `"italic"`. La valeur par défaut est `"normal"`.
`fontWeight`	Les deux	Epaisseur de la police : `"none"` ou `"bold"`. La valeur par défaut est `"none"`. Tous les composants peuvent également accepter la valeur `"normal"` au lieu de `"none"` pendant un appel à la méthode `setStyle()`, mais les appels suivants à `getStyle()` renvoient `"none"`.
`textAlign`	Les deux	Alignement du texte : `"left"`, `"right"` ou `"center"`. La valeur par défaut est `"left"`.
`textDecoration`	Les deux	Décoration du texte : `"none"` ou `"underline"`. La valeur par défaut est `"none"`.
`textIndent`	Les deux	Nombre indiquant le retrait du texte. La valeur par défaut est 0.

Le composant Alert inclut trois catégories de texte différentes. La configuration des propriétés de texte du composant Alert lui-même fournit les valeurs par défaut des trois catégories, comme dans l'exemple ci-dessous :

```
import mx.controls.Alert;
_global.styles.Alert.setStyle("color", 0x000099);
Alert.show("This is a test alert", "Title");
```

Pour définir les styles de texte d'une seule catégorie, le composant Alert fournit des propriétés statiques correspondant à des références à une occurrence de CSSStyleDeclaration.

Propriété statique	Texte affecté
`buttonStyleDeclaration`	Bouton
`messageStyleDeclaration`	Message
`titleStyleDeclaration`	Titre

L'exemple suivant montre comment mettre le titre d'un composant Alert en italique :

```
import mx.controls.Alert;
import mx.styles.CSSStyleDeclaration;

var titleStyles = new CSSStyleDeclaration();
titleStyles.setStyle("fontWeight", "bold");
titleStyles.setStyle("fontStyle", "italic");

Alert.titleStyleDeclaration = titleStyles;

Alert.show("Name is a required field", "Validation Error");
```

Les déclarations par défaut du style de titre définissent `fontWeight` sur « `bold` ». Si vous remplacez la propriété `titleStyleDeclaration`, cette valeur par défaut l'est également. Vous devez donc explicitement définir `fontWeight` sur « `bold` » si vous le souhaitez.

REMARQUE

Les styles de texte définis pour un composant Alert fournissent des styles par défaut à ses composants par héritage. Pour plus d'informations, reportez-vous à « Définition de styles hérités sur un conteneur » dans *Utilisation des composants ActionScript 2.0*.

Utilisation d'enveloppes avec le composant Alert

Le composant Alert étend le composant Window et utilise l'enveloppe d'arrière-plan du titre comme arrière-plan de titre, une occurrence de classe RectBorder pour sa bordure et des enveloppes Button pour les états visuels de ses boutons. Pour appliquer des enveloppes aux boutons et à la barre de titres lors de la programmation, modifiez les symboles Flash UI Components 2/Themes/MMDefault/Window Assets/Elements/TitleBackground et Flash UI Components 2/Themes/MMDefault/Button Assets/ButtonSkin. Pour plus d'informations, consultez « A propos de l'application d'enveloppes aux composants » dans *Utilisation des composants ActionScript 2.0*. La bordure et l'arrière-plan sont fournis par la classe RectBorder par défaut. Pour plus d'informations sur l'application d'enveloppes à la classe RectBorder, reportez-vous à « Classe RectBorder », à la page 1111.

Un composant Alert utilise les propriétés d'enveloppe suivantes pour appliquer une enveloppe dynamiquement aux boutons et à la barre de titre :

Propriété	Description	Valeur par défaut
`buttonUp`	Etat relevé des boutons.	`ButtonSkin`
`buttonUpEmphasized`	Etat relevé du bouton par défaut.	`ButtonSkin`
`buttonDown`	Etat enfoncé des boutons.	`ButtonSkin`
`buttonDownEmphasized`	Etat enfoncé du bouton par défaut.	`ButtonSkin`
`buttonOver`	Etat survolé des boutons.	`ButtonSkin`
`buttonOverEmphasized`	Etat survolé du bouton par défaut.	`ButtonSkin`
`titleBackground`	Barre de titre de la fenêtre.	`TitleBackground`

Pour définir le titre d'un composant Alert sur un symbole de clip personnalisé :

1. Sélectionnez Fichier > Nouveau, puis choisissez Fichier Flash (ActionScript 2.0).
2. Créez un nouveau symbole en choisissant Insertion > Nouveau symbole.
3. Nommez-le `TitleBackground`.
4. Si l'écran avancé n'est pas affiché, cliquez sur le bouton Avancé.
5. Sélectionnez Exporter pour ActionScript.

   L'identifiant est automatiquement renseigné avec `TitleBackground`.
6. Définissez la valeur de la classe sur `mx.skins.SkinElement`.

   SkinElement est une classe simple qui peut être utilisée pour tous les éléments d'enveloppe qui n'assurent pas leur propre implémentation dans ActionScript. Elle apporte les fonctionnalités de mouvement et de dimensionnement requises par les composants.

7. Assurez-vous que l'option Exporter dans la première image est bien activée, puis cliquez sur OK.
8. Ouvrez le nouveau symbole pour l'édition.
9. Servez-vous des outils de dessin pour créer une case rouge entourée d'une ligne noire.
10. Définissez le style de bordure sur Filet.
11. Définissez la case, et sa bordure, de sorte qu'elle soit positionnée au point (0,0), avec une largeur de 100 et une hauteur de 22.

    Le composant Alert définit alors la largeur appropriée pour l'enveloppe, mais utilise la hauteur existante pour le titre.
12. Cliquez sur le bouton Précédent pour revenir au scénario principal.
13. Faites glisser un composant Alert sur la scène, puis supprimez-le.

    Le composant Alert est ajouté à la bibliothèque, ce qui le rend disponible lors de l'exécution.
14. Pour créer un exemple d'occurrence Alert, ajoutez du code ActionScript au scénario principal.

```
import mx.controls.Alert;
Alert.show("This is a skinned Alert component","Title");
```

15. Choisissez Contrôle> Tester l'animation.

Classe Alert

Héritage MovieClip > Classe UIObject > Classe UIComponent > View > ScrollView > Composant Window > Alert

Nom de classe ActionScript mx.controls.Alert

Pour utiliser le composant Alert, vous devez faire glisser un composant Alert sur la scène, puis le supprimer afin qu'il soit présent dans la bibliothèque de documents sans apparaître dans l'application. Vous devez ensuite appeler `Alert.show()` pour afficher une fenêtre Alert. Vous pouvez transmettre des paramètres à `Alert.show()` pour ajouter un message, une barre de titre et des boutons dans la fenêtre Alert.

Dans la mesure où ActionScript est asynchrone, le composant Alert n'effectue pas de blocage, c'est-à-dire que les lignes de code ActionScript qui suivent l'appel à `Alert.show()` sont aussitôt exécutées. Vous devez ajouter des écouteurs pour gérer les événements `click` diffusés lorsque l'utilisateur clique sur un bouton, puis continuer votre code après la diffusion de l'événement.

REMARQUE	Dans des environnements d'exploitation qui opèrent un blocage (par exemple, Microsoft Windows), un appel à `Alert.show()` ne renvoie rien tant que l'utilisateur n'a pas effectué une action, par exemple cliqué sur un bouton.

Pour plus d'informations sur la classe Alert, reportez-vous à « Composant Window », à la page 1519 et à « Classe PopUpManager », à la page 1029.

Récapitulatif des méthodes de la classe Alert

Le tableau suivant présente la méthode de la classe Alert.

Méthode	Description
`Alert.show()`	Crée une fenêtre Alert avec des paramètres facultatifs.

Méthodes héritées de la classe UIObject

Le tableau suivant énumère les méthodes de la classe Alert héritées de la classe UIObject.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque des paramètres ont été définis dans les inspecteurs des propriétés et des composants.
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet jusqu'à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.

Méthode	Description
UIObject.setSkin()	Définit une enveloppe dans l'objet.
UIObject.setStyle()	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant énumère les méthodes de la classe Alert héritées de la classe UIComponent.

Méthode	Description
UIComponent.getFocus()	Renvoie une référence à l'objet ayant le focus.
UIComponent.setFocus()	Attribue le focus à l'occurrence de composant.

Méthodes héritées de la classe Window

Le tableau suivant énumère les méthodes de la classe Alert héritées de la classe Window.

Méthode	Description
Window.deletePopUp()	Supprime une occurrence de fenêtre créée par PopUpManager.createPopUp().

Récapitulatif des propriétés de la classe Alert

Le tableau suivant présente les propriétés de la classe Alert.

Propriété	Description
Alert.buttonHeight	Hauteur de chaque bouton, en pixels. La valeur par défaut est 22.
Alert.buttonWidth	Largeur de chaque bouton, en pixels. La valeur par défaut est 100.
Alert.CANCEL	Constante hexadécimale qui indique si la fenêtre Alert doit contenir un bouton Annuler.
Alert.cancelLabel	Texte d'étiquette du bouton Annuler.
Alert.NO	Constante hexadécimale qui indique si la fenêtre Alert doit contenir un bouton Non.
Alert.noLabel	Texte d'étiquette du bouton Non.
Alert.OK	Constante hexadécimale qui indique si la fenêtre Alert doit contenir un bouton OK.

Propriété	Description
`Alert.okLabel`	Texte d'étiquette du bouton OK.
`Alert.YES`	Constante hexadécimale qui indique si la fenêtre Alert doit contenir un bouton Oui.
`Alert.yesLabel`	Texte d'étiquette du bouton Oui.

Propriétés héritées de la classe UIObject

Le tableau suivant énumère les propriétés de la classe Alert héritées de la classe UIObject. Pour appeler ces propriétés à partir de l'objet Alert, utilisez le formulaire `Alert.`*`propertyName`*.

Propriété	Description
`UIObject.bottom`	Lecture seule. Position du bord inférieur de l'objet par rapport au bord inférieur de son parent.
`UIObject.height`	Lecture seule ; hauteur de l'objet, en pixels.
`UIObject.left`	Lecture seule ; bord gauche de l'objet, en pixels.
`UIObject.right`	Lecture seule ; position du bord droit de l'objet par rapport au bord droit de son parent.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Lecture seule ; renvoie la position du bord supérieur de l'objet par rapport à son parent.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Lecture seule ; largeur de l'objet, en pixels.
`UIObject.x`	Lecture seule ; bord gauche de l'objet, en pixels.
`UIObject.y`	Lecture seule ; bord supérieur de l'objet, en pixels.

Propriétés héritées de la classe UIComponent

Le tableau suivant énumère les propriétés de la classe Alert héritées de la classe UIComponent. Pour appeler ces propriétés à partir de l'objet Alert, utilisez le formulaire `Alert.propertyName`.

Propriété	Description
`UIComponent.enabled`	Indique si le composant peut recevoir le focus et la saisie.
`UIComponent.tabIndex`	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Propriétés héritées de la classe Window

Le tableau suivant énumère les propriétés de la classe Alert héritées de la classe Window.

Propriété	Description
`Window.closeButton`	Indique si la barre de titre comprend un bouton Fermer (`true`) ou non (`false`).
`Window.content`	Référence au contenu (clip racine) de la fenêtre.
`Window.contentPath`	Définit le nom du contenu à afficher dans la fenêtre.
`Window.title`	Texte qui apparaît dans la barre de titre.
`Window.titleStyleDeclaration`	Déclaration de style qui formate le texte dans la barre de titre.

Récapitulatif des événements de la classe Alert

Le tableau suivant présente un événement de la classe Alert.

Evénement	Description
`Alert.click`	Diffusé lorsque l'utilisateur clique sur un bouton dans une fenêtre Alert.

Evénements hérités de la classe UIObject

Le tableau suivant énumère les événements de la classe Alert hérités de la classe UIObject. Pour appeler ces événements à partir de l'objet Alert, utilisez le formulaire `Alert.eventName`.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant énumère les événements de la classe Alert hérités de la classe UIComponent. Pour appeler ces événements à partir de l'objet Alert, utilisez le formulaire `Alert.eventName`.

Evénement	Description
`UIComponent.focusIn`	Diffusé lorsqu'un objet reçoit le focus.
`UIComponent.focusOut`	Diffusé lorsqu'un objet perd le focus.
`UIComponent.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIComponent.keyUp`	Diffusé lorsqu'une touche est relâchée.

Evénements hérités de la classe Window

Le tableau suivant énumère les événements de la classe Alert hérités de la classe Window.

Evénement	Description
`Window.click`	Diffusé lorsque le bouton de fermeture est relâché.
`Window.complete`	Diffusé à la création d'une fenêtre.

Alert.buttonHeight

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
Alert.buttonHeight
```

Description

Propriété (classe) : propriété de classe (statique) qui modifie la hauteur des boutons. La valeur par défaut est 22.

Exemple

Lorsque la bibliothèque contient déjà un composant Alert, cet exemple redimensionne les boutons :

```
import mx.controls.Alert;

// Réglage des tailles de bouton.
Alert.buttonHeight = 50;
Alert.buttonWidth = 150;

// Affichage de la boîte de dialogue Alerte.
Alert.show("Launch Stock Application?", "Stock Price Alert", Alert.OK |
  Alert.CANCEL);
```

Voir aussi

Alert.buttonWidth

Alert.buttonWidth

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
Alert.buttonWidth
```

Description

Propriété (classe) : propriété de classe (statique) qui modifie la largeur des boutons. La valeur par défaut est 100.

Exemple

Si la bibliothèque contient déjà un composant Alert, ajoutez ce code ActionScript à la première image du scénario principal pour redimensionner les boutons :

```
import mx.controls.Alert;

// Réglage des tailles de bouton.
Alert.buttonHeight = 50;
Alert.buttonWidth = 150;

// Affichage de la boîte de dialogue Alerte.
Alert.show("Launch Stock Application?", "Stock Price Alert", Alert.OK |
  Alert.CANCEL);
```

Voir aussi

`Alert.buttonHeight`

Alert.CANCEL

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
Alert.CANCEL
```

Description

Propriété (constante) : propriété dont la valeur hexadécimale est 0x8. Cette propriété peut être utilisée pour le paramètre *`flags`* ou *`defaultButton`* de la méthode `Alert.show()`. Utilisée comme valeur du paramètre *`flags`*, elle indique qu'un bouton Annuler doit apparaître dans la fenêtre Alert. Si elle est utilisée comme valeur du paramètre *`defaultButton`*, le bouton Annuler a le focus initial et est déclenché lorsque l'utilisateur appuie sur Entrée (Windows) ou sur Retour (Macintosh). Si l'utilisateur passe à un autre bouton, ce dernier est déclenché lorsque l'utilisateur appuie sur Entrée.

Exemple

L'exemple suivant utilise `Alert.CANCEL` et `Alert.OK` comme valeurs du paramètre *`flags`* et affiche un composant Alert comportant un bouton OK et un bouton Annuler :

```
import mx.controls.Alert;
Alert.show("This is a generic Alert window", "Alert Test", Alert.OK |
  Alert.CANCEL, this);
```

Alert.cancelLabel

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
Alert.cancelLabel
```

Description

Propriété (classe) : propriété de classe (statique) qui indique l'étiquette de texte du bouton Annuler.

Exemple

L'exemple suivant définit l'étiquette du bouton Annuler sur « annulation » :

```
Alert.cancelLabel = "cancellation";
```

Alert.click

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
var clickHandler:Function = function(eventObject:Object) {
  // Insertion du code ici.
}
Alert.show(message[, title[, flags[, parent[, clickHandler[, icon[,
  defaultButton]]]]]])
```

Description

Evénement : diffusé à l'écouteur enregistré lorsque l'utilisateur clique sur le bouton OK, Oui, Non ou Annuler.

Les composants utilisent un modèle dispatcher(diffuseur)/écouteur d'événement.
Le composant Alert distribue un événement `click` si vous cliquez sur l'un de ses boutons, et l'événement est géré par une fonction (également appelée *gestionnaire*) sur un objet écouteur (*`listenerObject`*) que vous créez. Vous appelez la méthode `Alert.show()` et lui transmettez le nom du gestionnaire en tant que paramètre. Lorsque l'utilisateur clique sur un bouton de la fenêtre Alert, l'écouteur est appelé.

Lorsque l'événement survient, il transmet automatiquement un objet événement (*`eventObject`*) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés dans le code qui traitera l'événement. L'objet événement `Alert.click` possède une propriété `detail` supplémentaire dont la valeur est `Alert.OK`, `Alert.CANCEL`, `Alert.YES` ou `Alert.NO` en fonction du bouton sur lequel vous cliquez. Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

Lorsque la bibliothèque contient déjà un composant Alert, ajoutez ce code ActionScript à la première image du scénario principal pour créer un gestionnaire d'événements appelé `myClickHandler`. Le gestionnaire d'événements est transmis à la méthode `Alert.show()` comme cinquième paramètre. L'objet événement est capturé par le gestionnaire `myClickHandler` dans le paramètre `evt`. La propriété `detail` de l'objet événement est ensuite utilisée dans une instruction `trace` pour envoyer le nom du bouton sur lequel l'utilisateur a cliqué (`Alert.OK` ou `Alert.CANCEL`) vers le panneau Sortie.

```
import mx.controls.Alert;

// Définition des actions du bouton.
var myClickHandler:Function = function (evt_obj:Object) {
 switch (evt_obj.detail) {
 case Alert.OK :
  trace("You clicked: " + Alert.okLabel);
  break;
 case Alert.CANCEL :
  trace("You clicked: " + Alert.cancelLabel);
  break;
 }
};

// Affichage de la boîte de dialogue.
Alert.show("This is a test of errors", "Error", Alert.OK | Alert.CANCEL,
  this, myClickHandler);
```

Alert.NO

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
Alert.NO
```

Description

Propriété (constante) : propriété dont la valeur hexadécimale est 0x2. Cette propriété peut être utilisée pour le paramètre *flags* ou *defaultButton* de la méthode `Alert.show()`. Utilisée comme valeur du paramètre *flags*, elle indique qu'un bouton Non doit apparaître dans la fenêtre Alert. Si elle est utilisée comme valeur du paramètre *defaultButton*, le bouton Annuler a le focus initial et est déclenché lorsque l'utilisateur appuie sur Entrée (Windows) ou sur Retour (Macintosh). Si l'utilisateur passe à un autre bouton, ce dernier est déclenché lorsque l'utilisateur appuie sur Entrée.

Exemple

L'exemple suivant utilise `Alert.NO` et `Alert.YES` comme valeurs du paramètre *flags* et affiche un composant Alert comportant un bouton Non et un bouton Oui :

```
import mx.controls.Alert;
Alert.show("This is a generic Alert window", "Alert Test", Alert.NO |
  Alert.YES, this);
```

Alert.noLabel

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
Alert.noLabel
```

Description

Propriété (classe) : propriété de classe (statique) qui indique l'étiquette de texte du bouton Non.

Exemple

L'exemple suivant définit l'étiquette du bouton Non sur « nyet » :

```
Alert.noLabel = "nyet";
```

Alert.NONMODAL

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
Alert.NONMODAL
```

Description

Propriété (constante) ; propriété dont la valeur hexadécimale est 0x8000. Cette propriété peut être utilisée pour le paramètre *flags* de la méthode `Alert.show()`. Cette propriété indique qu'une fenêtre Alert devrait être non modale pour permettre aux utilisateurs d'interagir avec les boutons et les occurrences situées sous la fenêtre affichée. Par défaut, les fenêtres générées avec `Alert.show()` sont modales, ce qui signifie que les utilisateurs ne peuvent pas cliquer sur n'importe quel élément, sauf sur la fenêtre actuellement ouverte.

Exemple

L'exemple suivant affiche deux occurrences du composant Button sur la scène. Cliquez sur un bouton pour ouvrir une fenêtre modale et empêcher l'utilisateur de cliquer sur les boutons tant que la fenêtre Alert n'est pas fermée. Le deuxième bouton ouvre une fenêtre non modale qui permet à l'utilisateur de continuer à cliquer sur les boutons situés sous la fenêtre Alert non modale actuellement ouverte. Pour tester cet exemple, ajoutez des occurrences du composant Alert et du composant Button à la bibliothèque du document en cours et ajoutez le code suivant à l'image 1 du scénario principal :

```
import mx.controls.Alert;

this.createClassObject(mx.controls.Button, "modal_button", 10, {_x:10,
  _y:10});
this.createClassObject(mx.controls.Button, "nonmodal_button", 20, {_x:120,
  _y:10});
```

```
modal_button.label = "modal";
modal_button.addEventListener("click", modalListener);
function modalListener(evt_obj:Object):Void {
  var a:Alert = Alert.show("This is a modal Alert window", "Alert Test",
  Alert.OK, this);
  a.move(100, 100);
}

nonmodal_button.label = "nonmodal";
nonmodal_button.addEventListener("click", nonmodalListener);
function nonmodalListener(evt_obj:Object):Void {
  var a:MovieClip = Alert.show("This is a nonmodal Alert window", "Alert
  Test", Alert.OK | Alert.NONMODAL, this);
  a.move(100, 100);
}
```

Alert.OK

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`Alert.OK`

Description

Propriété (constante) : propriété dont la valeur hexadécimale est 0x4. Cette propriété peut être utilisée pour le paramètre *`flags`* ou *`defaultButton`* de la méthode `Alert.show()`. Utilisée comme valeur du paramètre *`flags`*, elle indique qu'un bouton OK doit apparaître dans la fenêtre Alert. Si elle est utilisée comme valeur du paramètre *`defaultButton`*, le bouton OK a le focus initial et est déclenché lorsque l'utilisateur appuie sur Entrée (Windows) ou sur Retour (Macintosh). Si l'utilisateur passe à un autre bouton, ce dernier est déclenché lorsque l'utilisateur appuie sur Entrée.

Exemple

L'exemple suivant utilise `Alert.OK` et `Alert.CANCEL` comme valeurs du paramètre *`flags`* et affiche un composant Alert comportant un bouton OK et un bouton Annuler :

```
import mx.controls.Alert;
Alert.show("This is a generic Alert window", "Alert Test", Alert.OK |
  Alert.CANCEL, this);
```

Alert.okLabel

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
Alert.okLabel
```

Description

Propriété (classe) : propriété de classe (statique) qui indique l'étiquette de texte du bouton OK.

Exemple

L'exemple suivant définit l'étiquette du bouton OK sur « okay » :

```
Alert.okLabel = "okay";
```

Alert.show()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
Alert.show(message[, title[, flags[, parent[, clickHandler[, icon[,
  defaultButton]]]]]])
```

Paramètres

`message` Message à afficher.

`title` Texte de la barre de titre dans la fenêtre Alert. Ce paramètre est facultatif. S'il n'est pas défini, la barre de titre est vide.

`flags` Paramètre facultatif qui indique les boutons devant apparaître dans la fenêtre Alert. La valeur par défaut est `Alert.OK` qui permet d'afficher un bouton OK. Si vous utilisez plusieurs valeurs, séparez-les par un caractère `|`. Utilisez une ou plusieurs des valeurs suivantes : `Alert.OK`, `Alert.CANCEL`, `Alert.YES`, `Alert.NO`.

Vous pouvez également utiliser `Alert.NONMODAL` pour indiquer que la fenêtre Alert n'est pas modale. Une fenêtre non modale permet à l'utilisateur d'interagir avec d'autres fenêtres dans l'application.

`parent` Fenêtre parent du composant Alert. La fenêtre Alert se centre elle-même dans la fenêtre parent. Utilisez la valeur `null` ou `undefined` pour désigner le scénario _root. La fenêtre parent doit être une sous-classe de la classe UIComponent, que ce soit comme autre composant Flash ou fenêtre personnalisée (pour plus d'informations, reportez-vous à « Présentation de l'héritage » du guide *Formation à ActionScript 2.0 dans Adobe Flash)*. Ce paramètre est facultatif.

`clickHandler` Gestionnaire des événements `click` diffusés lorsque l'utilisateur clique sur les boutons. Outre les propriétés d'objet événement click standard, il existe une propriété `detail` qui contient la valeur flag du bouton sur lequel l'utilisateur a cliqué (`Alert.OK`, `Alert.CANCEL`, `Alert.YES`, `Alert.NO`). Ce gestionnaire peut être une fonction ou un objet. Pour plus d'informations, consultez « Gestion des événements à l'aide d'écouteurs » dans *Utilisation des composants ActionScript 2.0*.

`icon` Chaîne correspondant à l'identifiant de liaison d'un symbole de la bibliothèque. Ce symbole est utilisé comme icône s'affichant à gauche du texte d'alerte. Ce paramètre est facultatif.

`defaultButton` Désigne le bouton qui dispose du focus initial et qui est activé lorsque l'utilisateur appuie sur Entrée (Windows) ou Retour (Macintosh). Si l'utilisateur passe à un autre bouton par tabulation, ce dernier est déclenché lorsque l'utilisateur appuie sur Entrée.

Ce paramètre peut prendre l'une des valeurs suivantes : `Alert.OK`, `Alert.CANCEL`, `Alert.YES`, `Alert.NO`.

Valeur renvoyée

L'occurrence Alert créée.

Description

Méthode (classe) : méthode (statique) de classe qui affiche une fenêtre Alert contenant un message et des éléments facultatifs (titre, boutons et icône). Le titre de la fenêtre Alert apparaît en haut de la fenêtre et est aligné sur la gauche. L'icône apparaît à gauche du texte du message. Les boutons sont centrés sous le texte du message et l'icône.

Exemple

Le code suivant est un exemple simple de fenêtre Alert modale contenant un bouton OK :

```
mx.controls.Alert.show("Hello, world!");
```

Le code suivant définit un gestionnaire click qui envoie un message au panneau Sortie indiquant les boutons sur lesquels l'utilisateur a cliqué. (La bibliothèque doit contenir un composant Alert pour que ce code affiche une alerte ; pour ajouter le composant à la bibliothèque, faites-le glisser sur la scène puis supprimez-le) :

```
import mx.controls.Alert;

// Définition des actions du bouton.
var myClickHandler:Function = function (evt_obj:Object) {
 if (evt_obj.detail == Alert.OK) {
  trace(Alert.okLabel);
 } else if (evt_obj.detail == Alert.CANCEL) {
  trace(Alert.cancelLabel);
 }
};

// Affichage de la boîte de dialogue.
var dialog_obj:Object = Alert.show("Test Alert", "Test", Alert.OK |
  Alert.CANCEL, null, myClickHandler, "testIcon", Alert.OK);
```

Alert.YES

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
Alert.YES
```

Description

Propriété (constante) : propriété dont la valeur hexadécimale est 0x1. Cette propriété peut être utilisée pour le paramètre *flags* ou *defaultButton* de la méthode `Alert.show()`. Utilisée comme valeur du paramètre *flags*, elle indique qu'un bouton Oui doit apparaître dans la fenêtre Alert. Si elle est utilisée comme valeur du paramètre *defaultButton*, le bouton Oui a le focus initial et est déclenché lorsque l'utilisateur appuie sur Entrée (Windows) ou sur Retour (Macintosh). Si l'utilisateur passe à un autre bouton, ce dernier est déclenché lorsque l'utilisateur appuie sur Entrée.

Exemple

L'exemple suivant utilise `Alert.NO` et `Alert.YES` comme valeurs du paramètre *`flags`* et affiche un composant Alert comportant un bouton Non et un bouton Oui :

```
import mx.controls.Alert;
Alert.show("This is a generic Alert window", "Alert Test", Alert.NO |
  Alert.YES, this);
```

Alert.yesLabel

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
Alert.yesLabel
```

Description

Propriété (classe) : propriété de classe (statique) qui indique l'étiquette de texte du bouton Oui.

Exemple

L'exemple suivant définit l'étiquette du bouton Oui sur « da » :

```
Alert.yesLabel = "da";
```

CHAPITRE 4

Composant Button

4

Le composant Button est un bouton d'interface utilisateur rectangulaire dont les dimensions sont modifiables. Vous pouvez ajouter une icône personnalisée à un bouton. Vous pouvez également modifier son comportement pour transformer un bouton-poussoir en bouton bascule. Un bouton bascule reste enfoncé lorsque l'utilisateur clique dessus, puis reprend l'état relevé au clic suivant.

REMARQUE

Un composant Button est pris en charge pour ActionScript 2.0 et ActionScript 3.0. Ce document présente le composant de la version 2. Si vous utilisez celui de la version 3, reportez-vous à « Utilisation du composant Button » dans *Utilisation des composants ActionScript 3.0*.

Un bouton peut être activé ou désactivé dans une application. En état désactivé, un bouton ne réagit pas aux commandes de la souris ou du clavier. Un bouton activé reçoit le focus si vous cliquez sur son entrée ou si vous appuyez sur la touche de tabulation pour l'atteindre. Lorsque l'occurrence d'un bouton a le focus, vous pouvez la contrôler à l'aide des touches suivantes :

Touche	Description
Contrôle +	Place le focus sur l'objet précédent.
Espace	Sélectionne ou libère le composant et déclenche l'événement `click`.
	Place le focus sur l'objet suivant.

Pour plus d'informations sur le contrôle du focus, reportez-vous à « Classe FocusManager », à la page 745 ou à « Création de la navigation personnalisée du focus » dans *Utilisation des composants ActionScript 2.0*.

L'aperçu en direct des occurrences de bouton reflète les modifications apportées aux paramètres dans l'inspecteur Propriétés ou l'inspecteur des composants pendant la programmation. Cependant, dans l'aperçu en direct, une icône personnalisée est représentée sur la scène par un carré gris.

Lorsque vous ajoutez le composant Button à une application, vous pouvez utiliser le panneau Accessibilité pour le rendre accessible aux lecteurs d'écran. Commencez par ajouter la ligne de code suivante :

```
mx.accessibility.ButtonAccImpl.enableAccessibility();
```

Quel que soit le nombre d'occurrences d'un composant, l'activation de son accessibilité ne se fait qu'une fois.

Utilisation du composant Button

Les boutons sont les éléments de base de tous les formulaires et de toutes les applications Web. Chaque fois que l'utilisateur doit pouvoir déclencher un événement, vous utilisez des boutons. Par exemple, la plupart des formulaires comportent un bouton Envoyer. Vous pouvez également ajouter des boutons Précédent et Suivant à une présentation.

Pour ajouter une icône à un bouton, vous devez sélectionner ou créer le clip ou le symbole graphique à utiliser comme icône. Ce symbole doit être enregistré au point 0,0 pour une mise en forme appropriée sur le bouton. Sélectionnez le symbole de l'icône dans le panneau Bibliothèque, ouvrez la boîte de dialogue Liaison dans le menu contextuel et saisissez un identifiant de liaison. Il s'agit de la valeur à entrer pour le paramètre de l'icône dans l'inspecteur Propriétés ou l'inspecteur des composants. Vous pouvez également entrer cette valeur pour la propriété `Button.icon` d'ActionScript.

REMARQUE	Lorsque l'icône est plus grande que le bouton, elle dépasse ses bordures.

Pour désigner un bouton comme bouton-poussoir par défaut dans une application (bouton qui reçoit l'événement click lorsque l'utilisateur appuie sur Entrée), utilisez `FocusManager.defaultPushButton`.

Paramètres du composant Button

Dans l'inspecteur Propriétés ou l'inspecteur de composants (option de menu Fenêtre > Inspecteur de composants), vous pouvez définir les paramètres de création suivants pour chaque occurrence Button :

icon Ajoute une icône personnalisée au bouton. La valeur est l'identifiant de liaison d'un clip ou d'un symbole graphique dans la bibliothèque ; il n'existe pas de valeur par défaut.

label Définit le texte inscrit sur le bouton. La valeur par défaut est Button.

labelPlacement Oriente le texte de l'étiquette sur le bouton par rapport à l'icône. Ce paramètre peut prendre l'une des quatre valeurs suivantes : `left`, `right`, `top` ou `bottom`. La valeur par défaut est `right`. Pour plus d'informations, voir `Button.labelPlacement`.

selected Si le paramètre toggle est `true`, ce paramètre spécifie si le bouton est enfoncé (`true`) ou relâché (`false`). La valeur par défaut est `false`.

toggle Transforme le bouton en bouton bascule. Si la valeur est `true`, le bouton reste enfoncé lorsque l'on clique sur son entrée et reprend l'état relevé au clic suivant. Si la valeur est `false`, le bouton se comporte comme un bouton-poussoir normal. La valeur par défaut est `false`.

Dans l'inspecteur des composants, vous pouvez définir les paramètres supplémentaires suivants pour chaque occurrence du composant Button (Fenêtre > Inspecteur de composants) :

enabled est une valeur booléenne qui indique si le composant peut recevoir le focus et l'entrée. La valeur par défaut est `true`.

visible est une valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`). La valeur par défaut est `true`.

REMARQUE

Les propriétés minHeight et minWidth sont utilisées par des routines de dimensionnement internes. Elles sont définies dans UIObject et remplacées par différents composants selon vos besoins. Vous pouvez utiliser ces propriétés si vous apportez une disposition personnalisée à votre application. Dans le cas contraire, la définition de ces propriétés dans l'inspecteur de composants n'a aucun effet visible.

Vous pouvez rédiger du code ActionScript pour contrôler ces options et d'autres options du composant Button à l'aide des propriétés, méthodes et événements ActionScript. Pour plus d'informations, voir « Classe Button », à la page 103.

Création d'une application avec le composant Button

La procédure suivante décrit l'ajout d'un composant Button à une application lors de la programmation. Dans cet exemple, le bouton affiche un message lorsque vous cliquez dessus.

Pour créer une application avec le composant Button :

1. Sélectionnez Fichier > Nouveau, puis choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser un composant Button du panneau Composants vers la scène.
3. Dans l'inspecteur Propriétés, procédez comme suit :
   - Entrez le nom d'occurrence **my_button**.
   - Entrez **Click me** pour le paramètre label.
   - Entrez **BtnIcon** comme paramètre de l'icône (icon).

     Pour utiliser une icône, le clip ou le symbole graphique doit être stocké dans la bibliothèque avec un identifiant de liaison, à utiliser comme paramètre icon. Dans cet exemple, l'identifiant de liaison est BtnIcon.
   - Définissez la propriété `toggle` sur `true`.
4. Choisissez l'image 1 dans le scénario, ouvrez le panneau Actions et saisissez le code suivant :

   ```
   function clicked(){
     trace("You clicked the button!");
   }
   my_button.addEventListener("click", clicked);
   ```

   La dernière ligne de code appelle une fonction de gestionnaire d'événements `clicked` pour l'événement « click ». Elle utilise la méthode « EventDispatcher.addEventListener() », à la page 517 avec une fonction personnalisée pour gérer l'événement.
5. Choisissez Contrôle > Tester l'animation.
6. Lorsque vous cliquez sur le bouton, Flash affiche le message « You clicked the button! » (Vous avez cliqué sur le bouton !).

Pour créer un composant Button à l'aide d'ActionScript :

1. Sélectionnez Fichier > Nouveau, puis choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser le composant Button du panneau Composants vers la bibliothèque du document en cours.

   Cette opération permet d'ajouter le composant à la bibliothèque, mais elle ne le rend pas visible dans l'application.

3. Dans la première image du scénario principal, ajoutez le code ActionScript suivant au panneau Actions pour créer une occurrence du composant Button :

```
this.createClassObject(mx.controls.Button, "my_button", 10,
  {label:"Click me"});
my_button.move(20, 20);
```

La méthode `UIObject.createClassObject()` permet de créer l'occurrence du composant Button nommée **my_button** et de spécifier une propriété d'étiquette. Le code utilise ensuite la méthode `UIObject.move()` pour positionner le bouton.

4. Ajoutez maintenant le code ActionScript suivant pour créer un écouteur d'événements et une fonction de gestionnaire d'événements :

```
function clicked() {
  trace("You clicked the button!");
}
my_button.addEventListener("click", clicked);
```

Il utilise la méthode « EventDispatcher.addEventListener() », à la page 517 avec une fonction personnalisée pour gérer l'événement.

5. Choisissez Contrôle > Tester l'animation.
6. Lorsque vous cliquez sur le bouton, Flash affiche le message « You clicked the button! » (Vous avez cliqué sur le bouton !).

Lorsque vous utilisez le composant Button avec d'autres composants, vous pouvez créer des fonctions de gestionnaire d'événements plus complexes. Dans cet exemple, l'événement « click » provoque la modification de l'affichage des panneaux par le composant Accordion.

Pour utiliser un événement Button avec un autre composant :

1. Sélectionnez Fichier > Nouveau, puis choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser le composant Button du panneau Composants jusqu'à la bibliothèque du document actif.

   Cette opération permet d'ajouter le composant à la bibliothèque, mais elle ne permet pas de le rendre visible dans l'application.
3. Faites glisser le composant Accordion du panneau Composants à la bibliothèque du document actuel.

4. Dans la première image du scénario principal, ajoutez le code ActionScript suivant au panneau Actions pour créer une occurrence du composant Button :

```
this.createClassObject(mx.containers.Accordion, "my_acc", 0);
my_acc.move(10, 40);
my_acc.createChild(mx.core.View, "panelOne", {label: "Panel One"});
my_acc.createChild(mx.core.View, "panelTwo", {label: "Panel Two"});

this.createClassObject(mx.controls.Button, "panelOne_button", 10,
  {label:"Panel One"});
panelOne_button.move(10, 10);
this.createClassObject(mx.controls.Button, "panelTwo_button", 20,
  {label:"Panel Two"});
panelTwo_button.move(120, 10);
```

   Cette procédure utilise la méthode `UIObject.createClassObject()` pour créer les occurrences des composants Button et Accordion. Le code utilise ensuite la méthode `UIObject.move()` pour positionner les occurrences.

5. Ajoutez maintenant le code ActionScript suivant pour créer des écouteurs d'événements et des fonctions de gestionnaire d'événements :

```
// Création du gestionnaire pour l'événement button.
function changePanel(evt_obj:Object):Void {
 // Modification de l'affichage du composant Accordion en fonction
 // du bouton sélectionné.
 switch (evt_obj.target._name) {
 case "panelOne_button" :
  my_acc.selectedIndex = 0;
  break;
 case "panelTwo_button" :
  my_acc.selectedIndex = 1;
  break;
 }
}

// Ajout d'écouteurs pour les boutons.
panelOne_button.addEventListener("click", changePanel);
panelTwo_button.addEventListener("click", changePanel);
```

   La méthode `EventDispatcher.addEventListener()` est utilisée avec une fonction personnalisée pour gérer les événements.

6. Sélectionnez Contrôle > Tester l'animation.

7. Lorsque vous cliquez sur un bouton, le composant Accordion change le panneau actuel.

Personnalisation du composant Button

Vous pouvez orienter un composant Button dans le sens horizontal et vertical pendant la création et à l'exécution. Lors de la programmation, sélectionnez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez la méthode `setSize()` (reportez-vous à `UIObject.setSize()`) ou des propriétés et des méthodes applicables de la classe Button (reportez-vous à « Classe Button », à la page 103). Le redimensionnement du bouton n'affecte pas la taille de l'icône ni celle de l'étiquette.

Le cadre de sélection d'une occurrence de bouton est invisible et désigne également la zone active de l'occurrence. Si vous augmentez la taille de l'occurrence, vous augmentez également la taille de la zone active. Si le cadre de sélection est trop petit pour contenir l'étiquette, celle-ci est découpée à la bonne taille.

Lorsque l'icône est plus grande que le bouton, elle dépasse ses bordures.

Utilisation de styles avec le composant Button

Vous pouvez définir des propriétés de style pour modifier l'aspect de l'occurrence d'un bouton. Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, consultez « Utilisation de styles pour personnaliser la couleur et le texte des composants » dans *Utilisation des composants ActionScript 2.0*.

Un composant Button prend en charge les styles suivants :

Style	Thème	Description
`themeColor`	Halo	Couleur de base d'un composant. `"haloGreen"`, `"haloBlue"` et `"haloOrange"` sont des valeurs possibles. La valeur par défaut est `"haloGreen"`.
`backgroundColor`	Sample	Couleur d'arrière-plan. La valeur par défaut est 0xEFEBEF (gris clair). Le thème Halo utilise 0xF8F8F8 (gris très clair) comme couleur d'arrière-plan du bouton lorsque ce dernier est relevé et `themeColor` lorsqu'il est enfoncé. Seule la couleur d'arrière-plan du bouton relevé peut être modifiée dans le thème Halo, en appliquant une enveloppe au bouton. Voir « Utilisation des enveloppes avec le composant Button », à la page 97.

Style	Thème	Description
`borderStyle`	Sample	Le composant Button utilise une occurrence RectBorder comme bordure dans le thème Sample et répond au styles définis dans cette classe. Voir « Classe RectBorder », à la page 1111. Avec le thème Halo, le composant Button utilise une bordure arrondie personnalisée dont les couleurs ne sont pas modifiables, à l'exception de `themeColor`.
`color`	Les deux	Couleur du texte. La valeur par défaut est 0x0B333C pour le thème Halo et vide pour le thème Sample.
`disabledColor`	Les deux	Couleur du texte lorsque le composant est désactivé. La couleur par défaut est 0x848384 (gris foncé).
`embedFonts`	Les deux	Valeur booléenne qui indique si la police spécifiée dans `fontFamily` est de type intégré. Ce style doit être défini sur `true` si `fontFamily` fait référence à une police intégrée. Dans le cas contraire, la police intégrée n'est pas utilisée. Si ce style est défini sur `true` et que `fontFamily` ne fait pas référence à une police intégrée, aucun texte n'est affiché. La valeur par défaut est `false`.
`fontFamily`	Les deux	Nom de la police du texte. La valeur par défaut est `"_sans"`.
`fontSize`	Les deux	Taille, en points, de la police. La valeur par défaut est 10.
`fontStyle`	Les deux	Style de police : `"normal"` ou `"italic"`. La valeur par défaut est `"normal"`.
`fontWeight`	Les deux	Epaisseur de la police : « `none` » ou « `bold` ». La valeur par défaut est `"none"`. Tous les composants peuvent également accepter la valeur `"normal"` au lieu de `"none"` pendant un appel à la méthode `setStyle()`, mais les appels suivants à `getStyle()` renvoient `"none"`.
`textDecoration`	Les deux	Décoration du texte : `"none"` ou `"underline"`. La valeur par défaut est `"none"`.

Utilisation des enveloppes avec le composant Button

En réponse aux bordures et icônes des 16 différents états possibles, le composant Button comprend 32 enveloppes différentes personnalisables. Pour appliquer une enveloppe au composant Button pendant la programmation, créez de nouveaux symboles de clip avec les graphiques désirés, puis définissez les identifiants de liaison des symboles à l'aide d'ActionScript. (Pour plus d'informations, voir « Utilisation d'ActionScript pour dessiner les enveloppes du composant Button », à la page 99.)

L'implémentation par défaut des enveloppes Button fournies avec les thèmes Halo et Sample utilise l'API de dessin ActionScript pour dessiner les états des boutons et le symbole de clip associé à une classe ActionScript pour fournir toutes les enveloppes au composant Button.

Le composant Button dispose de nombreuses enveloppes et d'une bordure et d'une icône pour chaque état, car ses états sont également très nombreux. Quatre propriétés et l'interaction de l'utilisateur contrôlent l'état d'une occurrence Button. Les propriétés suivantes affectent les enveloppes :

Propriété	Description
`emphasized`	Donne deux apparences différentes aux occurrences Button et sert généralement à les mettre en surbrillance, par exemple le bouton par défaut dans un formulaire.
`enabled`	Indique si le bouton autorise ou non une interaction de l'utilisateur.
`toggle`	Présente une valeur sélectionnée et non sélectionnée et utilise des enveloppes différentes pour signaler la valeur en cours. Dans le cas d'une occurrence Button dont la propriété `toggle` est définie sur `false`, les enveloppes `false` sont utilisées. Lorsque cette propriété est définie sur `true`, l'enveloppe dépend de la propriété `selected`.
`selected`	Lorsque la propriété `toggle` est définie sur `true`, elle détermine si le composant Button est sélectionné (`true` ou `false`). Les différentes enveloppes permettent d'identifier la valeur et sont, par défaut, le seul moyen de représenter cette valeur à l'écran.

Si un bouton est activé, il affiche l'état Dessus lorsque le pointeur de la souris est placé au-dessus. Le bouton reçoit le focus d'entrée et affiche l'état Abaissé lorsque l'utilisateur clique dessus. Le bouton retrouve l'état Dessus lorsque la souris est relâchée. Si le pointeur s'éloigne du bouton alors que la souris est enfoncée, le bouton retrouve son état d'origine et garde le focus d'entrée. Si le paramètre toggle est défini sur `true`, l'état du bouton ne change pas jusqu'à ce que le bouton de la souris soit relâché sur lui.

Si un bouton est désactivé, il affiche son état désactivé, quelle que soit l'interaction de l'utilisateur.

Les composants Button prennent en charge les propriétés d'enveloppe suivantes :

Propriété	Description
falseUpSkin	Etat relevé (normal).
falseDownSkin	Etat enfoncé.
falseOverSkin	Etat survolé.
falseDisabledSkin	Etat désactivé.
trueUpSkin	Etat basculé.
trueDownSkin	Etat enfoncé-basculé.
trueOverSkin	Etat survolé-basculé.
trueDisabledSkin	Etat désactivé-basculé.
falseUpSkinEmphasized	Etat relevé (normal) d'un bouton mis en relief.
falseDownSkinEmphasized	Etat enfoncé d'un bouton mis en relief.
falseOverSkinEmphasized	Etat survolé d'un bouton mis en relief.
falseDisabledSkinEmphasized	Etat désactivé d'un bouton mis en relief.
trueUpSkinEmphasized	Etat basculé d'un bouton mis en relief.
trueDownSkinEmphasized	Etat enfoncé-basculé d'un bouton mis en relief.
trueOverSkinEmphasized	Etat survolé-basculé d'un bouton mis en relief.
trueDisabledSkinEmphasized	Etat désactivé-basculé d'un bouton mis en relief.
falseUpIcon	Etat relevé de l'icône.
falseDownIcon	Etat enfoncé de l'icône.
falseOverIcon	Etat survolé de l'icône.
falseDisabledIcon	Etat désactivé de l'icône.
trueUpIcon	Etat basculé de l'icône.
trueOverIcon	Etat survolé-basculé de l'icône.
trueDownIcon	Etat enfoncé-basculé de l'icône.
trueDisabledIcon	Etat désactivé-basculé de l'icône.
falseUpIconEmphasized	Etat relevé de l'icône d'un bouton mis en relief.
falseDownIconEmphasized	Etat enfoncé de l'icône d'un bouton mis en relief.
falseOverIconEmphasized	Etat survolé de l'icône d'un bouton mis en relief.
falseDisabledIconEmphasized	Etat désactivé de l'icône d'un bouton mis en relief.
trueUpIconEmphasized	Etat basculé de l'icône d'un bouton mis en relief.

Propriété	Description
`trueOverIconEmphasized`	Etat survolé-basculé de l'icône d'un bouton mis en relief.
`trueDownIconEmphasized`	Etat enfoncé-basculé de l'icône d'un bouton mis en relief.
`trueDisabledIconEmphasized`	Etat désactivé-basculé de l'icône d'un bouton mis en relief.

La valeur par défaut de toutes les propriétés d'enveloppe se terminant par « Skin » est `ButtonSkin` et celle de toutes les propriétés « Icon » est `undefined`. Les propriétés présentant un suffixe « Skin » fournissent un arrière-plan et une bordure, alors que celles qui présentent un suffixe « Icon » fournissent une petite icône.

Outre les enveloppes d'icône, le composant Button reconnaît également une propriété `icon` standard. La différence entre la propriété standard et la propriété de style est que cette dernière vous permet de définir des icônes pour les états individuels alors que la première ne permet de définir qu'une icône appliquée à tous les états. Lorsque la propriété `icon` et des propriétés de styles sont définies pour une occurrence Button, le comportement de celle-ci peut être imprévisible.

Reportez-vous à *Utilisation des composants ActionScript 2.0* dans l'Aide pour regarder une démo interactive présentant l'utilisation de chaque enveloppe.

Utilisation d'ActionScript pour dessiner les enveloppes du composant Button

Les enveloppes par défaut des thèmes Halo et Sample utilisent le même élément d'enveloppe pour tous les états et tracent les graphiques réels via ActionScript. L'implémentation de Halo utilise une extension de la classe RectBorder et un code de dessin personnalisé pour tracer les états. L'implémentation de Sample utilise la même enveloppe et la même classe ActionScript que le thème Halo, différentes propriétés étant définies dans ActionScript pour modifier l'apparence du composant Button.

Pour créer une classe ActionScript, l'utiliser comme enveloppe et fournir différents états, l'enveloppe peut lire la propriété de style `borderStyle` de l'enveloppe et la propriété `emphasized` du parent pour déterminer l'état. Le tableau suivant présente le style de bordure défini pour chaque enveloppe :

Propriété	Style de bordure
`falseUpSkin`	`falseup`
`falseDownSkin`	`falsedown`
`falseOverSkin`	`falserollover`
`falseDisabled`	`falsedisabled`

Propriété	Style de bordure
trueUpSkin	trueup
trueDownSkin	truedown
trueOverSkin	truerollover
trueDisabledSkin	truedisabled

Pour créer une enveloppe de bouton ActionScript personnalisée :

1. Sélectionnez Fichier > Nouveau, puis choisissez Fichier ActionScript.
2. Copiez le code ActionScript suivant dans le fichier :

```
import mx.skins.RectBorder;
import mx.core.ext.UIObjectExtensions;

class RedGreenBlueSkin extends RectBorder
{
  static var symbolName:String = "RedGreenBlueSkin";
  static var symbolOwner:Object = RedGreenBlueSkin;

  function size():Void
  {
    var c:Number; // couleur
    var borderStyle:String = getStyle("borderStyle");

    switch (borderStyle) {
      case "falseup":
      case "falserollover":
      case "falsedisabled":
        c = 0x7777FF;
        break;
      case "falsedown":
        c = 0x77FF77;
        break;
      case "trueup":
      case "truedown":
      case "truerollover":
      case "truedisabled":
        c = 0xFF7777;
        break;
    }

    clear();
    var thickness = _parent.emphasized ? 2 : 0;
    lineStyle(thickness, 0, 100);
    beginFill(c, 100);
    drawRect(0, 0, __width, __height);
    endFill();
  }
```

```
    // Obligatoire pour les enveloppes.
    static function classConstruct():Boolean
    {
      UIObjectExtensions.Extensions();
      _global.skinRegistry["ButtonSkin"] = true;
      return true;
    }
    static var classConstructed:Boolean = classConstruct();
    static var UIObjectExtensionsDependency = UIObjectExtensions;
}
```

Cette classe crée une case carrée en fonction du style de bordure : une case bleue pour les états false relevé, survolé et désactivé, une case verte pour l'état enfoncé normal et une case rouge pour l'enfant développé. Elle dessine une bordure en filet pour l'état normal et une bordure épaisse lorsque le bouton est mis en relief.

3. Enregistrez le fichier.

   Dans cet exemple, nommez le fichier RedGreenBlueSkin.as.

4. Sélectionnez Fichier > Nouveau, puis choisissez Fichier Flash (ActionScript 2.0).
5. Enregistrez le fichier dans le même dossier que le fichier AS.
6. Créez un symbole en choisissant Insertion > Nouveau symbole.
7. Nommez-le `ButtonSkin`.
8. Si l'écran avancé n'est pas affiché, cliquez sur le bouton Avancé.
9. Sélectionnez Exporter pour ActionScript.

   L'identifiant est automatiquement renseigné avec `ButtonSkin`.

10. Définissez la valeur de la classe sur `RedGreenBlueSkin`.
11. Assurez-vous que l'option Exporter dans la première image est bien activée, puis cliquez sur OK.
12. Faites glisser un composant Button sur la scène.
13. Sélectionnez Contrôle > Tester l'animation.

Utilisation de clips pour personnaliser les enveloppes du composant Button

L'exemple ci-dessus décrit l'utilisation d'une classe ActionScript pour personnaliser l'enveloppe d'un composant Button, méthode utilisée par les enveloppes des thèmes Halo et Sample. Toutefois, cet exemple utilise de simples cases colorées et il est plus judicieux dans ce cas d'employer des symboles de clip différents comme enveloppes.

Pour créer des symboles de clip pour les enveloppes de bouton :

1. Sélectionnez Fichier > Nouveau, puis choisissez Fichier Flash (ActionScript 2.0).
2. Créez un symbole en choisissant Insertion > Nouveau symbole.
3. Nommez-le `RedButtonSkin`.
4. Si l'écran des paramètres avancés n'est pas affiché, cliquez sur le bouton Avancé.
5. Sélectionnez Exporter pour ActionScript.

   L'identifiant est automatiquement renseigné avec `RedButtonSkin`.
6. Définissez la valeur de la classe sur `mx.skins.SkinElement`.
7. Assurez-vous que l'option Exporter dans la première image est bien activée, puis cliquez sur OK.
8. Ouvrez le nouveau symbole pour l'édition.
9. Servez-vous des outils de dessin pour créer une case rouge entourée d'une ligne noire.
10. Définissez le style de bordure sur Filet.
11. Définissez la case, et sa bordure, de sorte qu'elle soit positionnée au point (0,0) avec une largeur et une hauteur de 100.

    Au besoin, la classe `SkinElement` redimensionne le contenu.
12. Répétez les étapes 2 à 11 et créez des enveloppes verte et bleue, nommées BlueButtonSkin et GreenButtonSkin.
13. Cliquez sur le bouton Précédent pour revenir au scénario principal.
14. Faites glisser un composant Button sur la scène.
15. Définissez la valeur de la propriété `toggled` sur `true` pour voir les trois enveloppes.
16. L'occurrence Button étant sélectionnée, copiez le code ActionScript suivant dans le panneau Actions.

```
onClipEvent(initialize) {
  falseUpSkin = "BlueButtonSkin";
  falseDownSkin = "GreenButtonSkin";
  falseOverSkin = "BlueButtonSkin";
  falseDisabledSkin = "BlueButtonSkin";
  trueUpSkin = "RedButtonSkin";
  trueDownSkin = "RedButtonSkin";
  trueOverSkin = "RedButtonSkin";
  trueDisabledSkin = "RedButtonSkin";
}
```

17. Choisissez Contrôle > Tester l'animation.

Classe Button

Héritage MovieClip > Classe UIObject > Classe UIComponent > Classe SimpleButton > Button

Nom de classe ActionScript mx.controls.Button

Les propriétés de la classe Button permettent d'effectuer les opérations suivantes au moment de l'exécution : ajouter une icône à un bouton, créer une étiquette de texte et indiquer si le bouton doit se comporter comme un bouton bascule ou un bouton-poussoir.

La définition d'une propriété de la classe Button avec ActionScript annule le paramètre du même nom défini dans l'inspecteur Propriétés ou des composants.

Le composant Button utilise le gestionnaire de focus pour remplacer le rectangle de focus par défaut de Flash Player et tracer un rectangle de focus personnalisé aux coins arrondis. Pour plus d'informations, consultez « Création de la navigation personnalisée du focus » dans *Utilisation des composants ActionScript 2.0*.

Toutes les classes de composants ont une propriété `version` qui correspond à une propriété de classe. Les propriétés de classe ne sont disponibles que pour la classe elle-même. La propriété `version` renvoie une chaîne qui indique la version du composant. Pour accéder à cette propriété, utilisez le code suivant :

```
trace(mx.controls.Button.version);
```

REMARQUE

Le code `trace(myButtonInstance.version);` renvoie `undefined`.

La classe du composant Button diffère de l'objet Button ActionScript intégré.

Méthodes de la classe Button

La classe Button ne présente pas de méthodes exclusives.

Méthodes héritées de la classe UIObject

Le tableau suivant énumère les méthodes de la classe Button héritées de la classe UIObject. Lors de l'appel à ces méthodes depuis l'objet Button, utilisez le formulaire *buttonInstance.methodName*.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet jusqu'à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
`UIObject.setSize()`	Redimensionne l'objet à la taille demandée.
`UIObject.setSkin()`	Définit une enveloppe dans l'objet.
`UIObject.setStyle()`	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant énumère les méthodes de la classe Button héritées de la classe UIComponent. Lors de l'appel à ces méthodes depuis l'objet Button, utilisez le formulaire *buttonInstance.methodName*.

Méthode	Description
`UIComponent.getFocus()`	Renvoie une référence à l'objet ayant le focus.
`UIComponent.setFocus()`	Attribue le focus à l'occurrence de composant.

Propriétés de la classe Button

Le tableau suivant présente les propriétés de la classe Button.

Propriété	Description
`Button.icon`	Spécifie une icône pour une occurrence de bouton.
`Button.label`	Spécifie le texte qui apparaît dans un bouton.
`Button.labelPlacement`	Spécifie l'orientation du texte de l'étiquette par rapport à une icône.

Propriétés héritées de la classe SimpleButton

Le tableau suivant énumère les propriétés de la classe Button héritées de la classe SimpleButton. Lorsque vous accédez à ces propriétés, utilisez le formulaire *`buttonInstance.propertyName`*.

Propriété	Description
`SimpleButton.emphasized`	Indique si un bouton a l'aspect d'un bouton-poussoir par défaut.
`SimpleButton.emphasizedStyleDeclaration`	Déclaration de style lorsque la propriété `emphasized` est définie sur `true`.
`SimpleButton.selected`	Valeur booléenne indiquant si le bouton est sélectionné (`true`) ou non (`false`). La valeur par défaut est `false`.
`SimpleButton.toggle`	Valeur booléenne indiquant si le comportement du bouton est celui d'un bouton bascule (`true`) ou non (`false`). La valeur par défaut est `false`.

Propriétés héritées de la classe UIObject

Le tableau suivant énumère les propriétés de la classe Button héritées de la classe UIObject. Lors de l'appel à ces propriétés depuis l'objet Button, utilisez le formulaire *`buttonInstance.propertyName`*.

Propriété	Description
`UIObject.bottom`	Lecture seule ; position du bord inférieur de l'objet par rapport au bord inférieur de son parent.
`UIObject.height`	Lecture seule ; hauteur de l'objet, en pixels.
`UIObject.left`	Lecture seule ; bord gauche de l'objet, en pixels.

Propriété	Description
`UIObject.right`	Lecture seule ; position du bord droit de l'objet par rapport au bord droit de son parent.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Lecture seule ; renvoie la position du bord supérieur de l'objet par rapport à son parent.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Lecture seule ; largeur de l'objet, en pixels.
`UIObject.x`	Lecture seule ; bord gauche de l'objet, en pixels.
`UIObject.y`	Lecture seule ; bord supérieur de l'objet, en pixels.

Propriétés héritées de la classe UIComponent

Le tableau suivant énumère les propriétés de la classe Button héritées de la classe UIComponent. Lors de l'appel à ces propriétés depuis l'objet Button, utilisez le formulaire *`buttonInstance.propertyName`*.

Propriété	Description
`UIComponent.enabled`	Indique si le composant peut recevoir le focus et la saisie.
`UIComponent.tabIndex`	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Evénements de la classe Button

La classe Button ne présente pas d'événements exclusifs.

Evénements hérités de la classe SimpleButton

Le tableau suivant énumère les événements de la classe Button hérités de la classe SimpleButton.

Propriété	Description
`SimpleButton.click`	Diffusé lorsque l'utilisateur clique sur un bouton.

Evénements hérités de la classe UIObject

Le tableau suivant énumère les événements de la classe Button hérités de la classe UIObject.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant énumère les événements de la classe Button hérités de la classe UIComponent.

Evénement	Description
`UIComponent.focusIn`	Diffusé lorsqu'un objet reçoit le focus.
`UIComponent.focusOut`	Diffusé lorsqu'un objet perd le focus.
`UIComponent.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIComponent.keyUp`	Diffusé lorsqu'une touche est relâchée.

Button.icon

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

buttonInstance.icon

Description

Propriété : chaîne spécifiant l'identifiant de liaison d'un symbole dans la bibliothèque. Cet identifiant sera utilisé comme icône pour l'occurrence d'un bouton. L'icône peut être un symbole de clip ou un symbole graphique avec un point d'alignement supérieur gauche. Vous devez redimensionner le bouton si l'icône est trop grande car ces deux éléments ne seront pas redimensionnés automatiquement. Lorsque l'icône est plus grande que le bouton, elle dépasse automatiquement ses bordures.

Pour créer une icône personnalisée, vous devez créer un clip ou un symbole graphique. Sélectionnez le symbole sur la scène en mode de modification des symboles et saisissez 0 dans les cases X et Y de l'inspecteur Propriétés. Dans le panneau Bibliothèque, sélectionnez le clip, puis Liaison dans le menu contextuel Bibliothèque. Activez l'option Exporter pour ActionScript et saisissez un identifiant dans le champ Identifiant.

La valeur par défaut est une chaîne vide (`""`) qui indique qu'il n'y a pas d'icône.

Utilisez la propriété `labelPlacement` pour définir la position de l'icône par rapport au bouton.

REMARQUE	Dans Flash, l'icône n'apparaît pas sur la scène. Pour visualiser l'icône, choisissez Contrôle > Tester l'animation.

Exemple

Lorsqu'un bouton se trouve sur la scène avec le nom d'occurrence `my_button`, le code suivant affecte le clip du panneau Bibliothèque avec l'identifiant de liaison `happiness` à l'occurrence du composant Button comme une icône :

```
my_button.icon = "happiness";
```

Vous pouvez également créer le bouton et affecter l'icône entièrement dans ActionScript à l'aide de la méthode `UIObject.createClassObject()` (vous devez avoir créé une icône correspondant au bouton avec l'identifiant de liaison `happiness`). Faites d'abord glisser le composant Button du panneau Composants vers la bibliothèque du document en cours pour que le composant apparaisse dans la bibliothèque, mais pas sur la scène. Ajoutez ensuite le code ActionScript suivant dans la première image du scénario principal :

```
this.createClassObject(mx.controls.Button, "my_button", 1, {icon:
  "happiness"});
```

Voir aussi

`Button.labelPlacement`

Button.label

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
buttonInstance.label
```

Description

Propriété : spécifie l'étiquette de texte pour une occurrence de bouton. Par défaut, l'étiquette apparaît centrée sur le bouton. L'appel de cette méthode remplace le paramètre d'étiquette défini lors de la programmation, spécifié dans l'inspecteur Propriétés ou des composants. La valeur par défaut est `"Button"`.

Exemple

Lorsqu'un bouton se trouve sur la scène avec le nom d'occurrence `my_button`, le code suivant définit l'étiquette sur « Test Button :

```
my_button.label = "Test Button";
```

Vous pouvez également créer le bouton et affecter l'étiquette entièrement dans ActionScript à l'aide de la méthode `UIObject.createClassObject()`. Faites d'abord glisser le composant Button du panneau Composants jusqu'à la bibliothèque du document actif, ainsi le composant apparaît dans la bibliothèque, mais pas sur la scène. Ajoutez ensuite le code ActionScript suivant dans la première image du scénario principal :

```
this.createClassObject(mx.controls.Button, "my_button", 1, {label: "Test
  Button"});
```

Voir aussi

`Button.labelPlacement`

Button.labelPlacement

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
buttonInstance.labelPlacement
```

Description

Propriété : définit la position de l'étiquette par rapport à l'icône. La valeur par défaut est `"right"`. Quatre valeurs sont possibles, l'icône et l'étiquette étant toujours centrées verticalement ou horizontalement dans la zone de délimitation du bouton :

- `"right"` L'étiquette est placée à droite de l'icône.
- `"left"` L'étiquette est placée à gauche de l'icône.
- `"bottom"` L'étiquette est placée sous l'icône.
- `"top"` L'étiquette est placée au-dessus de l'icône.

Exemple

Lorsqu'un bouton se trouve sur la scène avec le nom d'occurrence `my_button` et que le panneau Bibliothèque contient un symbole avec l'identifiant de liaison `happiness`, le code suivant définit l'alignement de l'étiquette à gauche de l'icône :

```
my_button.icon = "happiness";
my_button.label = "Test Button";
my_button.labelPlacement = "left";
```

Vous pouvez également créer le bouton et définir l'alignement de l'étiquette entièrement dans ActionScript à l'aide de la méthode `UIObject.createClassObject()`. Faites d'abord glisser le composant Button du panneau Composants à la bibliothèque du document actuel de façon à ce qu'il apparaisse dans la bibliothèque mais pas sur la scène. Ajoutez ensuite le code ActionScript suivant dans la première image du scénario principal :

```
this.createClassObject(mx.controls.Button, "my_button", 1, {label: "Test
  Button", icon: "happiness", labelPlacement: "left"});
```

CHAPITRE 5

API CellRenderer

5

L'API CellRenderer est un ensemble de propriétés et de méthodes utilisé par les composants de type liste (List, DataGrid, Tree, Menu et ComboBox) pour manipuler et afficher le contenu de cellules personnalisées dans chacune de leurs lignes. Cette cellule personnalisée peut contenir un composant prédéfini, le composant CheckBox par exemple, ou toute classe que vous créez.

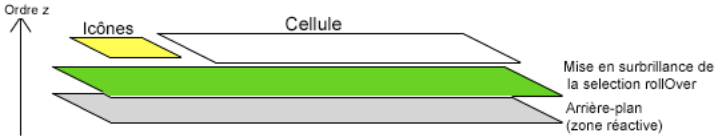
REMARQUE	L'API CellRenderer est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Présentation de la classe List

Pour utiliser l'API CellRenderer, vous devez bien connaître la classe List. Les composants DataGrid, Tree, Menu et ComboBox étant des extensions de la classe List, une bonne connaissance de cette classe permet de bien comprendre leur fonctionnement.

Composition du composant List

Les composants List sont constitués de lignes. Ces lignes affichent les mises en valeur de sélection et de survol, traduisent l'état « cliquable » de la sélection et jouent un rôle essentiel dans le défilement. Outre les icônes et les mises en valeur de sélection (icônes de nœud, flèches d'agrandissement d'un composant Tree, par exemple), une ligne est composée d'une ou de plusieurs cellules, dans le cas du composant DataGrid. Par défaut, ces cellules sont des objets TextField qui implémentent l'API CellRenderer. Cependant, vous pouvez indiquer à une liste d'utiliser une classe de composant différente pour chaque cellule (ligne). La seule condition est que la classe implémente l'API CellRenderer, car le composant List l'utilise pour communiquer avec la cellule.



Ordre d'empilement d'une ligne dans un composant List ou DataGrid

REMARQUE

Si une cellule contient des gestionnaires d'événements de bouton (`onPress`, etc.), il se peut que la zone réactive d'arrière-plan ne reçoive pas les entrées nécessaires pour déclencher les événements.

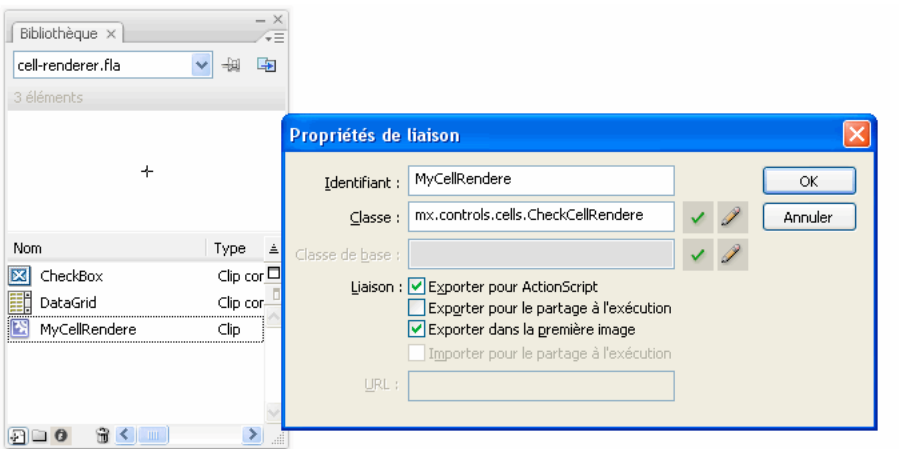
A propos du comportement de défilement du composant List

La classe List utilise un algorithme relativement complexe pour le défilement. Une liste contient autant de lignes qu'il est possible d'afficher simultanément. Les éléments excédant la valeur de la propriété `rowCount` n'obtiennent aucune ligne. Lors du défilement, la liste déplace toutes les lignes vers le haut ou le bas (en fonction de la direction choisie). Elle recycle ensuite les lignes hors de vue, les réinitialise et les utilise comme nouvelles lignes du défilement. Pour ce faire, la liste définit la valeur de l'ancienne ligne sur le nouvel élément et déplace l'ancienne ligne jusqu'à l'endroit où apparaît ce nouvel élément pendant le défilement.

Du fait de ce comportement de défilement, une même cellule ne contiendra pas qu'une seule valeur. Le recyclage des lignes implique que l'objet CellRenderer doit savoir comment réinitialiser entièrement son état lorsqu'il prend une nouvelle valeur. Par exemple, si votre CellRenderer crée une icône permettant d'afficher un élément, il peut avoir besoin de supprimer cette icône au moment d'afficher le rendu d'un nouvel élément. Supposons que votre objet CellRenderer est un conteneur qui sera alimenté par de nombreuses valeurs d'éléments au fil du temps, et qu'il doit savoir comment se modifier entièrement pour passer de l'affichage d'une valeur à une autre. En réalité, votre cellule doit même savoir comment rendre correctement des éléments non définis, ce qui peut signifier la suppression de l'intégralité de l'ancien contenu de la cellule.

Utilisation de l'API CellRenderer

Vous devez écrire une classe avec quatre méthodes (`CellRenderer.getPreferredHeight()`, `CellRenderer.getPreferredWidth()`, `CellRenderer.setSize()` et `CellRenderer.setValue()`) que le composant basé sur des listes utilisera pour communiquer avec la cellule (si la classe étend les fonctions de la classe UIObject, vous pouvez utiliser `size()` et non `CellRenderer.setSize()`). La classe doit être définie dans le champ de texte Classe de la boîte de dialogue Propriétés de liaison d'un symbole de clip dans votre application Flash.



A titre d'exemple, observez la classe CheckCellRenderer qui implémente l'API CellRenderer. Elle est située dans /Documents and Settings/*utilisateur*/Local Settings/Application Data/Adobe/Flash CS3/en/Configuration/Classes/mx/controls/cells. Reportez-vous également à la documentation du composant DataGrid pour consulter les informations sur CellRenderer, notamment « Stratégies de performance DataGrid », à la page 269.

Deux méthodes et une propriété (`CellRenderer.getCellIndex()`, `CellRenderer.getDataLabel()` et `CellRenderer.listOwner`) sont automatiquement attribuées à une cellule pour lui permettre de communiquer avec le composant de listes. Supposons par exemple qu'une cellule contienne une case à cocher qui, lorsqu'elle est activée, entraîne la sélection d'une ligne. CellRenderer a besoin d'une référence au composant basé sur des listes qui le contient pour appeler la propriété `selectedIndex` de ce composant.
De même, cette API doit connaître l'élément d'index actuellement rendu pour pouvoir définir la propriété `selectedIndex` sur le nombre correct. Pour ce faire, elle peut utiliser les méthodes `CellRenderer.listOwner` et `CellRenderer.getCellIndex()`. Il n'est pas nécessaire d'implémenter ces éléments ActionScript car la cellule les reçoit automatiquement lorsqu'elle est placée dans le composant de listes.

Exemple simple de CellRenderer

Cette section présente un exemple de CellRenderer qui affiche plusieurs lignes de texte dans une cellule.

Le didacticiel suivant montre comment créer une classe CellRenderer affichant plusieurs lignes de texte dans les cellules d'un composant DataGrid.

Les fichiers complets, MultiLineCell.as et CellRenderer_tutorial.fla sont disponibles en ligne à l'adresse www.adobe.com/go/learn_fl_samples_fr.

Création de la classe CellRenderer MultiLineCell

Une classe CellRenderer doit implémenter les méthodes suivantes :

- `CellRenderer.getPreferredHeight()`
- `CellRenderer.getPreferredWidth()`

  La méthode `CellRenderer.getPreferredWidth()` est nécessaire uniquement pour les composants Menu ou DataGrid ; sinon, commentez-la hors code, comme indiqué dans l'exemple.
- `CellRenderer.setSize()`

  Si une classe CellRenderer étend les fonctions de la classe UIObject, utilisez à la place `implement size()`, comme indiqué dans l'exemple.
- `CellRenderer.setValue()`

Une classe CellRenderer doit également déclarer les méthodes et la propriété reçues de la classe List :

- `CellRenderer.getCellIndex()`
- `CellRenderer.getDataLabel()`
- `CellRenderer.listOwner`

Les étapes suivantes montrent comment créer un fichier de classe CellRenderer ActionScript 2, nommé **MultiLineCell.as**, et comment le lier à un nouveau symbole de clip dans un nouveau document Flash. Vous pouvez ensuite ajouter un composant DataGrid à la bibliothèque de document Flash. Sur la première image, ajoutez le code ActionScript qui crée de façon dynamique le composant DataGrid et affecte la classe MultiLineCell comme CellRenderer pour l'une de ses colonnes :

Pour créer la classe CellRenderer MultiLineCell :

1. Sélectionnez Fichier > Nouveau, puis choisissez Fichier ActionScript.
2. Enregistrez le fichier sous le nom **MultiLineCell.as**.
3. Entrez le code suivant dans le fichier MultiLineCell.as :

```
// ActionScript 2.0 class.
class MultiLineCell extends mx.core.UIComponent
{
  private var multiLineLabel; // Etiquette à utiliser pour le texte.
  private var owner; // Ligne qui contient cette cellule.
  private var listOwner; // Composant List, DataGrid ou Tree contenant
                         // cette cellule.

  // Décalage de la hauteur de cellule par rapport à la hauteur totale
  // de la ligne et largeur de cellule préférée.
  private static var PREFERRED_HEIGHT_OFFSET = 4;
  private static var PREFERRED_WIDTH = 100;
  // Profondeur de départ.
  private var startDepth:Number = 1;

  // Constructeur. Doit être vide.
  public function MultiLineCell()
  {
  }

  /* UIObject attend que vous renseigniez createChildren en instanciant
  tous les actifs du clip dont vous pouvez avoir besoin après
  l'initialisation. Dans ce cas, nous créons une étiquette*/
  public function createChildren():Void
  {
    // La méthode createLabel est une méthode utile de la classe
    // UIObject et constitue un
    // moyen pratique pour créer des étiquettes dans des composants.
```

```
    var c = multiLineLabel = this.createLabel("multiLineLabel",
  startDepth);
    // Lie le style de l'étiquette à celui de la grille.
    c.styleName = listOwner;
    c.selectable = false;
    c.tabEnabled = false;
    c.background = false;
    c.border = false;
    c.multiline = true;
    c.wordWrap = true;
  }

  public function size():Void
  {
/* En étendant UIComponent qui importe UIObject, vous obtenez
  automatiquement la méthode setSize ; cependant UIComponent attend que
  vous implémentiez size(). Supposons que __width et __height soient
  définis. Vous allez élargir la cellule pour l'adapter à rowHeight.
  rowHeight est une propriété du composant de type List dans lequel nous
  rendons une cellule. Dans la mesure où nous souhaitons que rowHeight
  s'adapte sur deux lignes, lors de la création du composant de type
  List à l'aide de cette classe CellRenderer, vérifiez que la propriété
  rowHeight est suffisamment large pour que deux lignes de texte
  puissent être rendues dedans.*/

/*__width et __height sont les variables sous-jacentes de lecture/
  définition .width et .height.*/
    var c = multiLineLabel;
    c.setSize(__width, __height);
  }

  // Indique la hauteur préférée de la cellule. Méthode héritée.
  public function getPreferredHeight():Number
  {
/* La cellule reçoit la propriété « owner », qui référence la ligne.
  Il est toujours préférable que la cellule corresponde à la hauteur de
  la ligne. Dans ce cas, nous conservons la cellule d'une taille
  légèrement inférieure.*/
    return owner.__height - PREFERRED_HEIGHT_OFFSET;
  }

  // Appelé par le propriétaire pour définir la valeur
  // de la cellule. Méthode héritée.
  public function setValue(suggestedValue:String, item:Object,
  selected:Boolean):Void
  {
```

```
/* Si l'élément n'est pas défini, rien ne doit être rendu dans la
  cellule. Définissez donc l'étiquette comme étant invisible. Remarque :
  pour faire défiler des composants de type List comme un composant
  DataGrid, les cellules doivent être vides dans la mesure où elles
  défilent hors de vue, puis elles sont réutilisées et définies sur une
  nouvelle valeur, ce qui produit un effet animé de défilement. C'est
  pourquoi, vous ne pouvez pas utiliser de cellule disposant toujours de
  données ou d'une même valeur.*/
    if (item==undefined){
      multiLineLabel.text._visible = false;
    }
    multiLineLabel.text = suggestedValue;
  }
  // fonction getPreferredWidth : uniquement pour les menus et
  // les en-têtes DataGrid
  // fonction getCellIndex : non utilisée pour cette classe CellRenderer
  // fonction getDataLabel : non utilisée pour cette classe CellRenderer
}
```

Création d'une application pour tester la classe de rendu de cellule MultiLineCell

Dans la procédure suivante, vous allez créer l'occurrence DataGrid et implémenter la classe MultiLineCell.

Pour créer une application avec un composant DataGrid utilisant la classe MultiLineCell :

1. Sélectionnez Fichier > Nouveau, puis choisissez Fichier Flash (ActionScript 2.0).
2. Choisissez Fichier > Enregistrer sous, nommez le fichier **cellRender_tutorial.fla**, puis enregistrez-le dans le dossier qui contient le fichier MultiLineCell.as.
3. Pour créer un symbole movieClip à lier à la classe MultiLineCell, choisissez Insertion > Nouveau symbole.
4. Dans la zone de texte Nom, tapez **MultiLineCell**.

   La valeur par défaut du Type est Animation. Laissez cette valeur sélectionnée.
5. Si l'écran des paramètres avancés n'est pas affiché, cliquez sur le bouton Avancé.
6. Dans la section Liaison, activez la case à cocher Exporter pour ActionScript.

   L'activation de cette option vous permet de joindre dynamiquement des occurrences de ce symbole à vos documents Flash pendant l'exécution. La zone de texte Identifiant affiche automatiquement MultiLineCell.

7. Définissez la Classe ActionScript 2.0 sur MultiLineCell (pour faire correspondre le nom de la classe MultiLineCell que vous avez créée précédemment).
8. Assurez-vous que l'option Exporter dans la première image est bien activée, puis cliquez sur OK.

REMARQUE	Si vous devez modifier les propriétés de liaison du symbole du clip MultiLineCell, cliquez avec le bouton droit sur le symbole dans la bibliothèque du document, puis sélectionnez Propriétés ou Liaison dans le menu contextuel.

9. Faites glisser le composant DataGrid du panneau Composants jusqu'à la bibliothèque.

   L'occurrence DataGrid est créée de façon dynamique par ActionScript à l'étape suivante.
10. Sélectionnez la première image du scénario principal (vérifiez que vous n'êtes pas encore en mode édition du clip MultiLineCell).
11. Dans le panneau Actions de la première image, entrez le code suivant pour créer de façon dynamique un composant DataGrid, lui affecter des données et affecter votre nouvelle classe CellRenderer :

```
// Création d'une occurrence du composant DataGrid.
this.createClassObject(mx.controls.DataGrid, "myGrid_dg", 1);

// Création d'un fournisseur de données pour la grille de
// données comportant quatre colonnes.
myDP = new Array();
var aLongString:String = "An example of a cell renderer class that
  displays a multiple line TextField";
myDP.addItem({firstName:"Winston", lastName:"Elstad", note:aLongString,
  item:100});
myDP.addItem({firstName:"Ric", lastName:"Dietrich", note:aLongString,
  item:101});
myDP.addItem({firstName:"Ewing", lastName:"Canepa", note:aLongString,
  item:102});
myDP.addItem({firstName:"Kevin", lastName:"Wade", note:aLongString,
  item:103});
myDP.addItem({firstName:"Kimberly", lastName:"Dietrich",
  note:aLongString, item:104});
myDP.addItem({firstName:"AJ", lastName:"Bilow", note:aLongString,
  item:105});
myDP.addItem({firstName:"Chuck", lastName:"Yushan", note:aLongString,
  item:106});
myDP.addItem({firstName:"John", lastName:"Roo", note:aLongString,
  item:107});
```

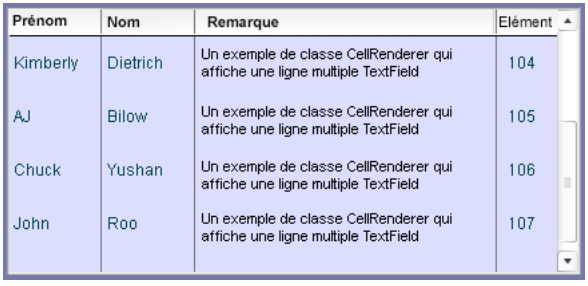
```
/* Affectation du fournisseur de données au composant DataGrid pour le
   remplir. Remarque : vous devez effectuer cette procédure avant
   d'appliquer CellRenderer. */
myGrid_dg.dataProvider = myDP;

/* Définition de certaines propriétés de grille basiques. Remarque : La
   hauteur de ligne de la grille de données doit refléter le nombre de
   lignes que vous souhaitez dans la classe CellRenderer MultiLineCell.
   Elle s'adapte à la hauteur de ligne. Pour la taille du texte par
   défaut, elle doit être approximativement de 40 pour 2 lignes ou 60
   pour 3 lignes.*/
myGrid_dg.setSize(430,200);
myGrid_dg.move(40,40);
myGrid_dg.rowHeight = 40; // Définition de 2 lignes de texte comme
                          // taille du texte par défaut.
myGrid_dg.getColumnAt(0).width = 70;
myGrid_dg.getColumnAt(1).width = 70;
myGrid_dg.getColumnAt(2).width = 220;
myGrid_dg.resizableColumns = true;
myGrid_dg.vScrollPolicy = "auto";
myGrid_dg.setStyle("backgroundColor", 0xD5D5FF);

// Affectation de la classe CellRenderer.
myGrid_dg.getColumnAt(2).cellRenderer = "MultiLineCell";
```

12. Enregistrez le document Flash, puis choisissez Contrôle > Tester l'animation.

Une grille de données s'affiche. Sa troisième colonne contient une cellule à plusieurs lignes.

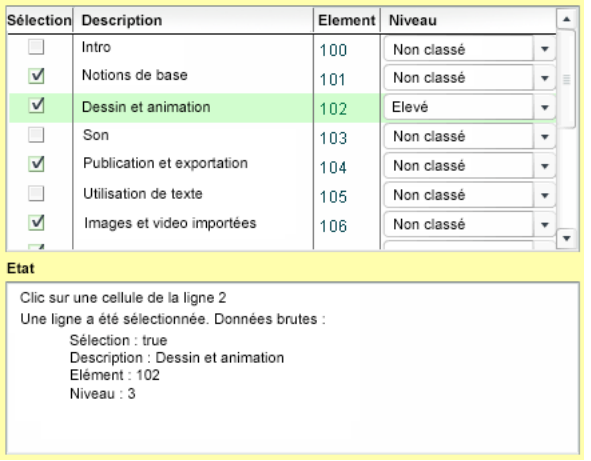


Prénom	Nom	Remarque	Elément
Kimberly	Dietrich	Un exemple de classe CellRenderer qui affiche une ligne multiple TextField	104
AJ	Bilow	Un exemple de classe CellRenderer qui affiche une ligne multiple TextField	105
Chuck	Yushan	Un exemple de classe CellRenderer qui affiche une ligne multiple TextField	106
John	Roo	Un exemple de classe CellRenderer qui affiche une ligne multiple TextField	107

Exemple de classe MultiLineCell complète.

Autres exemples de classes CellRenderer

D'autres exemples de classes CellRenderer affichant un composant ComboBox et un composant CheckCell sont également fournis. Les exemples sont disponibles en ligne à l'adresse www.adobe.com/go/learn_fl_samples_fr.



Autre exemple installé, nommé CellRenderers_Sample, et affichant un composant ComboBox et un composant CheckBox.

Méthodes à implémenter pour l'API CellRenderer

Pour que les composants List, DataGrid, Tree ou Menu puissent communiquer avec la cellule, vous devez écrire une classe avec les méthodes suivantes.

Méthode	Description
`CellRenderer.getPreferredHeight()`	Renvoie la hauteur préférée d'une cellule.
`CellRenderer.getPreferredWidth()`	Renvoie la largeur préférée d'une cellule.
`CellRenderer.setSize()`	Définit la largeur et la hauteur d'une cellule.
`CellRenderer.setValue()`	Définit le contenu à afficher dans la cellule.

Méthodes fournies par l'API CellRenderer

Les composants List, DataGrid, Tree et Menu fournissent les méthodes suivantes à la cellule lors de sa création dans l'un de ces composants. Il n'est donc pas nécessaire de les implémenter.

Méthode	Description
`CellRenderer.getCellIndex()`	Renvoie un objet avec deux champs, `columnIndex` et `itemIndex`, qui indiquent la position de la cellule.
`CellRenderer.getDataLabel()`	Renvoie une chaîne contenant le nom du champ de données du composant CellRenderer.

Propriétés fournies par l'API CellRenderer

Les composants List, DataGrid, Tree et Menu fournissent les propriétés suivantes à la cellule lors de sa création dans le composant. Il n'est donc pas nécessaire de les implémenter.

Propriété	Description
`CellRenderer.listOwner`	Référence au composant List contenant la cellule.
`CellRenderer.owner`	Référence à la ligne contenant la cellule.

CellRenderer.getCellIndex()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

componentInstance`.getCellIndex()`

Paramètres

Aucun.

Valeur renvoyée

Un objet comprenant deux champs : `columnIndex` et `itemIndex`.

Description

Méthode : renvoie un objet avec deux champs, `columnIndex` et `itemIndex` qui indiquent la position de la cellule dans le composant. Chaque champ est un nombre entier qui indique la position d'une colonne et d'un élément d'une cellule. Pour les composants autres que DataGrid, la valeur de `columnIndex` est toujours 0.

Cette méthode est fournie par la classe List. Il n'est donc pas nécessaire de l'implémenter. Déclarez-la comme suit dans votre classe CellRenderer et utilisez-la dans les fonctions de votre rendu de cellule.

```
var getCellIndex:Function;
```

Exemple

Cet exemple permet de modifier le fournisseur de données d'un composant DataGrid depuis une cellule.

```
var index = getCellIndex();
var colName = listOwner.getColumnAt(index.columnIndex).columnName;
listOwner.dataProvider.editField(index.itemIndex, colName, someVal);
```

CellRenderer.getDataLabel()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

componentInstance.getDataLabel()

Paramètres

Aucun.

Valeur renvoyée

Une chaîne.

Description

Méthode : renvoie une chaîne contenant le nom du champ de données du composant CellRenderer. Dans le cas du composant DataGrid, la méthode renvoie le nom de la colonne de la cellule en cours.

Cette méthode est fournie par la classe List. Il n'est donc pas nécessaire de l'implémenter. Déclarez-la comme suit dans votre classe CellRenderer et utilisez-la dans les fonctions de votre rendu de cellule.

```
var getDataLabel:Function;
```

Exemple

Le code suivant indique à la cellule le nom du champ de données dont elle effectue le rendu. Par exemple, si le nom du champ de données actuellement rendu par la cellule est « `Price` », la variable `p` correspond à présent à « `Price` ».

```
var p = getDataLabel();
```

CellRenderer.getPreferredHeight()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

componentInstance`.getPreferredHeight()`

Paramètres

Aucun.

Valeur renvoyée

La hauteur correcte de la cellule.

Description

Méthode qui renvoie la hauteur préférée d'une cellule. Ceci est particulièrement important pour obtenir la hauteur correcte du texte dans la cellule. Si vous définissez cette propriété sur une valeur supérieure à celle de la propriété `rowHeight` du composant, les cellules dépasseront au-dessus et au-dessous des lignes.

Cette méthode n'étant pas fournie par la classe List, vous devez l'implémenter. Elle indique aux lignes de la liste comment centrer la cellule et, au besoin, en ajuster la hauteur.
Si nécessaire, vous pouvez renvoyer une constante (par exemple 22), ou mesurer et renvoyer la hauteur du contenu. Vous pouvez également renvoyer `owner.height`, c'est-à-dire la hauteur de la ligne.

Exemple

Cet exemple renvoie la valeur 20, indiquant que la cellule doit avoir une hauteur de 20 pixels :

```
function getPreferredHeight(Void) :Number
{
   return 20;
}
```

Cet exemple renvoie une valeur inférieure de 4 pixels à la hauteur de la ligne :

```
function getPreferredHeight():Number
{
/* Vous savez que la cellule reçoit la propriété « owner », correspondant à
  la ligne. Il est toujours préférable que la cellule corresponde à la
  hauteur de la ligne.
*/
return owner.__height - 4;
}
```

CellRenderer.getPreferredWidth()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

componentInstance.getPreferredWidth()

Paramètres

Aucun.

Valeur renvoyée

Une valeur (numérique) qui indique la largeur correcte de la cellule.

Description

Méthode : largeur préférée d'une cellule. Si vous indiquez une largeur supérieure à celle du composant, la cellule peut être tronquée.

Implémentez cette méthode pour le composant Menu. La taille de votre cellule est ajustée à la largeur de la ligne, sauf dans un menu, qui doit mesurer le texte pour calculer la largeur de la ligne. Vous pouvez également implémenter cette méthode pour le composant DataGrid où l'objet headerRenderer vérifie qu'il faut ou non afficher la flèche permettant de trier.

Exemple

Cet exemple renvoie la valeur multipliée par 3, qui indique que la cellule doit être trois fois plus longue que la chaîne qu'elle affiche :

```
function getPreferredWidth():Number
{
   return myString.length*3;
}
```

Cet exemple commente la méthode `getPreferredWidth()` :

```
// fonction getPreferredWidth : uniquement pour un composant Menu
// ou DataGrid
```

CellRenderer.listOwner

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

componentInstance.listOwner

Description

Propriété : référence à la liste propriétaire de la cellule. Cette liste peut être un composant DataGrid, Tree, List ou Menu

Cette méthode est fournie par la classe List. Il n'est pas nécessaire de l'implémenter. Déclarez-la comme suit dans votre classe CellRenderer et utilisez-la comme référence pour revenir au composant List (ou Tree, Menu ou Grid) :

```
var listOwner:MovieClip; // ou UIObject, etc.
```

Exemple

Cet exemple trouve l'élément sélectionné de la liste dans une cellule :

```
var s = listOwner.selectedItem;
```

CellRenderer.owner

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
componentInstance.owner
```

Description

Propriété : référence à la ligne contenant la cellule.

Cette méthode est fournie par la classe List. Il n'est pas nécessaire de l'implémenter. Déclarez-la dans votre classe CellRenderer et utilisez-la comme référence.

```
var owner:MovieClip; // ou UIObject, etc.
```

CellRenderer.setSize()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
componentInstance.setSize(width, height)
```

Paramètres

width Nombre indiquant la largeur définie pour le positionnement du composant.

height Nombre indiquant la hauteur définie pour le positionnement du composant.

Valeur renvoyée

Aucune.

Description

Méthode qui permet à la liste d'indiquer à ses cellules la taille qu'elles doivent adopter. L'objet CellRenderer doit être positionné de façon à demeurer à l'intérieur de la zone spécifiée. Dans le cas contraire, la cellule risque de déborder dans d'autres parties de la liste et d'apparaître tronquée.

Si le rendu de la cellule étend la classe UIObject, implémentez plutôt la méthode `size()`. Ecrivez la même fonction que pour `setSize()`, mais servez-vous des propriétés `width` et `height` à la place des paramètres.

Exemple

Cet exemple dimensionne une image dans la cellule pour qu'elle reste dans les limites définies par la liste :

```
function setSize(w:Number, h:Number):Void
{
   image._width = w-2;
   image._height = h-2;
   image._x = image._y = 1;
}
```

Cet exemple étant une classe CellRenderer qui étend UIComponent (qui étend UIObject), vous devez implémenter `size()` à la place de `setSize()`, comme suit :

```
// En étendant UIComponent, vous obtenez du même coup setSize.
// Toutefois, UIComponent attend que vous implémentiez size().
// Supposons que __width et __height soient définis.
// Vous allez élargir la cellule pour l'adapter à rowHeight.

function size():Void
{
// __width et __height sont les variables sous-jacentes
// de lecture/définition .width et .height.
  var c = multiLineLabel;
  c._width = __width;
  c._height = __height;
}
```

CellRenderer.setValue()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

componentInstance.setValue(*suggested*, *item*, *selected*)

Paramètres

suggested Valeur à utiliser pour le texte de CellRenderer, le cas échéant.

item Objet correspondant à l'intégralité de l'élément à restituer. Le composant CellRenderer peut utiliser les propriétés de cet objet pour le rendu.

selected Chaîne dont les valeurs possibles sont : « `normal` », « `highlighted` » et « `selected` ».

Valeur renvoyée

Aucune.

Description

Méthode qui prend les valeurs données et en crée une représentation dans la cellule. Ceci annule les différences entre ce qui était affiché dans la cellule et ce qui doit être l'être dans la cellule pour le nouvel élément. (N'oubliez pas qu'une cellule peut afficher de nombreuses valeurs tout au long de son séjour dans la liste.) Il s'agit là de la méthode CellRenderer la plus importante et vous devez l'implémenter dans chaque composant CellRenderer.

La méthode `setValue()` est fréquemment appelée (par exemple, en cas de survol, de sélection, de redimensionnement d'une colonne ou de défilement). N'oubliez pas qu'une cellule risque de ne pas exister sur la scène et qu'elle ne doit pas toujours être mise à jour avec les données lorsque la méthode `setValue()` est appelée. A tout moment, une cellule particulière peut par exemple être défilée hors de la zone d'affichage ou elle risque d'être réutilisée pour restituer une autre valeur. C'est pourquoi, vous ne pouvez pas directement référencer une occurrence CellRenderer spécifique de la grille, et vous devez écrire les instructions `if` dans le corps de `setValue()` qui permettent l'exécution du code uniquement si le paramètre `item` est défini et qu'un changement s'est produit. Un paramètre item non défini indique que la cellule doit être visiblement vide et que la propriété `_visible` de tous les éléments de la cellule doit être définie sur `false`. Il peut être nécessaire que les cellules soient visiblement vides de façon temporaire, notamment lors d'un défilement du composant DataGrid.

Si une ligne est sélectionnée et que le pointeur de la souris est dessus, la valeur du paramètre sélectionné est « `highlighted` » et non pas « `selected` ». Cela devient un problème lorsque vous essayez d'obtenir un comportement de rendu différent selon l'état sélectionné ou non de la ligne. Pour tester l'état de la ligne en cours (sélectionné ou non), utilisez le code suivant :

```
var reallySelected:Boolean = selected != "normal" && listOwner.selectedNode
  == item;
```

Exemple

L'exemple suivant décrit l'utilisation de `setValue()` et de `editField()` pour référencer une occurrence CellRenderer dans une grille.

Etant donné qu'à tout moment une cellule peut ne pas exister sur la scène (elle peut être défilée hors de la zone d'affichage ou réutilisée pour restituer une autre valeur), vous ne pouvez pas directement référencer une occurrence CellRenderer dans la grille.

Pour communiquer avec une cellule spécifique de la grille, vous devez utiliser le fournisseur de données. Ce dernier gère toutes les informations d'états relatives à la grille. Pour pouvoir afficher une cellule donnée comme activée ou sélectionnée, le fournisseur de données doit comporter un champ correspondant pour détenir cette information. La méthode `setValue()` de votre composant CellRenderer communique les changements d'état du fournisseur de données à la cellule. Le code suivant implémente la méthode `setValue()` à partir d'un composant CellRenderer théorique qui affiche une case à cocher dans les cellules :

```
function setValue(str, itm, sel)
{
/* Supposons que le fournisseur de données possède deux champs appropriés
  pour cette cellule : cochée et activée.
Il pourrait se présenter sous la forme suivante :
[
{field1:"DisplayMe", field2:"SomeString", checked:true, enabled:false}
{field1:"DisplayMe", field2:"SomeString", checked:false, enabled:true}
{field1:"DisplayMe", field2:"SomeString", checked:true, enabled:true}
]
*/

/* Masquez tout ce qui est normalement rendu dans la cellule si item n'est
  pas défini. Sinon mettez à jour le contenu de la cellule avec les
  nouvelles données.
*/
  if (itm == undefined){
    myCheck._visible = false;
  }else{

    // Vérification de la redondance
    if (myCheck.selected!=itm.checked){
      myCheck.selected = itm.checked;
    }
    if (myCheck.enabled!=itm.enabled){
      myCheck.enabled = itm.enabled;
    }
  }
}
```

Pour activer la case à cocher de la seconde ligne, vous communiquez via le fournisseur de données. Toute modification apportée au fournisseur de données (par une méthode DataProvider comme `DataProvider.editField()`) appelle la méthode `setValue()` pour actualiser l'affichage de la grille. Ce code serait écrit dans une application Flash, dans une image, un objet ou un autre fichier de classe (mais pas dans le fichier de classe CellRenderer) :

```
// Nouvel appel de setValue()
myGrid.editField(1, "enabled", true);
```

L'exemple suivant permet de charger une image dans un composant Loader au sein de la cellule, en fonction de la valeur transmise :

```
function setValue(suggested, item, selected) : Void
{
/* Masquez tout ce qui est normalement rendu dans la cellule si item n'est
  pas défini. Sinon mettez à jour le contenu de la cellule avec les
  nouvelles données.
*/
  if (item == undefined){
    loader._visible = false;
  }else{
    // Suppression du composant Loader
    loader.contentPath = undefined;
    // Le fournisseur de données de la liste contient les URL
    // de différentes images
    if (suggested!=undefined){
      loader.contentPath = suggested;
    }
  }
}
```

L'exemple suivant provient d'un rendu de texte multiligne :

```
function setValue(suggested:String, item:Object, selected:Boolean):Void
{
/* Masquez tout ce qui est normalement rendu dans la cellule si item n'est
  pas défini. Sinon mettez à jour le contenu de la cellule avec les
  nouvelles données.
*/
  if (item == undefined){
    multiLineLabel._visible = false;
  }else{
    // Ajout du texte à l'étiquette.
    multiLineLabel.text = suggested;
  }
}
```

L'exemple suivant est tiré du rendu d'un composant Radio Button. Si le paramètre item n'est pas défini, la cellule peut être défilée hors de la zone d'affichage et doit être visiblement vide. Une instruction `if` permet de déterminer si le paramètre `item` n'est pas défini. S'il ne l'est pas, le bouton radio est masqué en définissant sa propriété `_visible` sur `false` ; dans le cas contraire, le bouton radio est mis à jour avec les nouvelles données et il s'affiche.

```
function setValue(str:String, item:Object, sel:String) : Void {
/* Masquez tout ce qui est normalement rendu dans la cellule si item n'est
  pas défini. Sinon mettez à jour le contenu de la cellule avec les
  nouvelles données.
*/
  if (item == undefined) {
    radio._visible = false; }
  else {
    trace(item.data + " " + item.label + " " + item.state + " " + sel);
    radio.label = item.label;
    radio.data = item.data;
    radio.selected = item.state;
    radio._visible = true;
  }
}
```

CHAPITRE 6

Composant CheckBox

6

Une case à cocher est un carré pouvant être activé ou désactivé. Lorsqu'elle est activée, une coche apparaît à l'intérieur. Vous pouvez lui ajouter une étiquette de texte et la placer à gauche, à droite, au-dessous ou au-dessus.

REMARQUE

Un composant CheckBox est pris en charge pour ActionScript 2.0 et ActionScript 3.0. Ce document présente le composant de la version 2. Si vous utilisez celui de la version 3, reportez-vous à « Utilisation du composant CheckBox » dans *Utilisation des composants ActionScript 3.0*.

Dans une application, les cases peuvent être activées ou désactivées. Lorsqu'une case est activée et qu'un utilisateur clique sur son entrée ou sur son étiquette, la case reçoit le focus d'entrée et son état enfoncé apparaît à l'écran. Si un utilisateur place le pointeur à l'extérieur du cadre de sélection d'une case ou de son étiquette en appuyant sur le bouton de la souris, l'aspect du composant revient à son état d'origine et conserve le focus d'entrée. L'état d'une case à cocher ne change pas tant que le bouton de la souris n'est pas relâché sur le composant. La case à cocher possède également deux états désactivés, sélectionné et désélectionné, qui ne permettent pas d'interagir avec la souris ou le clavier.

Lorsqu'une case est désactivée, elle affiche un aspect désactivé, quelle que soit l'interaction de l'utilisateur. En état désactivé, un bouton ne réagit pas aux commandes de la souris ou du clavier.

Une occurrence CheckBox reçoit le focus si un utilisateur clique sur son entrée ou utilise la touche de tabulation pour l'atteindre. Lorsqu'une occurrence de case à cocher a le focus, les touches suivantes permettent de le contrôler :

Touche	Description
Maj+Tab	Déplace le focus vers l'élément précédent.
Espace	Sélectionne ou désélectionne le composant et déclenche l'événement `click`.
Tab	Déplace le focus vers l'élément suivant.

Pour plus d'informations sur le contrôle du focus, reportez-vous à « Classe FocusManager », à la page 745 ou à « Création de la navigation personnalisée du focus » dans *Utilisation des composants ActionScript 2.0*.

L'aperçu en direct des occurrences CheckBox reflète les modifications apportées aux paramètres dans l'inspecteur des propriétés ou l'inspecteur des composants pendant la programmation.

Lorsque vous ajoutez le composant CheckBox à une application, vous pouvez utiliser le panneau Accessibilité pour le rendre accessible aux lecteurs d'écran. Commencez par ajouter la ligne de code suivante pour activer l'accessibilité :

```
mx.accessibility.CheckBoxAccImpl.enableAccessibility();
```

Quel que soit le nombre d'occurrences d'un composant, l'activation de son accessibilité ne se fait qu'une fois. Pour plus d'informations, reportez-vous à *Création de contenu accessible* dans *Utilisation de Flash*.

Utilisation du composant CheckBox

La case à cocher est l'un des éléments de base des formulaires et applications Web. Vous pouvez utiliser des cases à cocher là où vous voulez réunir un jeu de valeurs `true` ou `false` qui ne s'excluent pas réciproquement. Par exemple, un formulaire recueillant des informations personnelles sur un client peut comporter une liste de hobbies que le client doit sélectionner ; plusieurs loisirs peuvent être cochés.

Paramètres du composant CheckBox

Vous pouvez définir les paramètres de programmation suivants pour chaque composant CheckBox dans l'inspecteur des propriétés ou l'inspecteur des composants :

label définit la valeur du texte de la case à cocher. La valeur par défaut est `CheckBox`.

labelPlacement oriente le texte de l'étiquette sur la case à cocher. Ce paramètre peut prendre l'une des quatre valeurs suivantes : `left`, `right`, `top` ou `bottom`. La valeur par défaut est `right`. Pour plus d'informations, reportez-vous à `CheckBox.labelPlacement`.

selected définit la valeur initiale de la case à cocher sur activée (`true`) ou désactivée (`false`). La valeur par défaut est `false`.

Vous pouvez rédiger du code ActionScript pour contrôler ces options et d'autres options du composant CheckBox à l'aide des propriétés, méthodes et événements ActionScript. Pour plus d'informations, voir « Classe CheckBox », à la page 140.

Création d'une application avec le composant CheckBox

La procédure suivante décrit l'ajout d'un composant CheckBox à une application au cours de la programmation. L'exemple suivant est un formulaire à remplir pour s'inscrire dans une agence de rencontres en ligne. Le formulaire est une requête qui recherche des candidats susceptibles de convenir au client. Le formulaire de requête doit comporter une case intitulée « Restrict Age » pour permettre au client de limiter sa recherche à la tranche d'âge de son choix. Lorsque cette case à cocher est activée, le client peut saisir un âge minimal et maximal dans deux champs de texte. (Ces champs de texte ne sont actifs que lorsque la case est cochée.)

Pour créer une application avec le composant CheckBox :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser deux composants TextInput du panneau Composants sur la scène.
3. Dans l'inspecteur Propriétés, entrez les noms d'occurrences **minimumAge** et **maximumAge**.
4. Faites glisser un composant CheckBox du panneau Composants jusqu'à la scène.
5. Dans l'inspecteur des propriétés, procédez comme suit :
   - Entrez le nom d'occurrence **restrictAge**.
   - Entrez le paramètre d'étiquette **Restrict Age**.
6. Sélectionnez l'image 1 dans le scénario, ouvrez le panneau Actions et saisissez le code suivant :

```
var restrictAgeListener:Object = new Object();
restrictAgeListener.click = function (evt:Object) {
  minimumAge.enabled = evt.target.selected;
  maximumAge.enabled = evt.target.selected;
};
restrictAge.addEventListener("click", restrictAgeListener);
```

   Ce code crée un gestionnaire d'événements `click` qui active et désactive les composants des champs de texte `minimumAge` et `maximumAge`, déjà placés sur la scène. Pour plus d'informations, voir les sections `CheckBox.click` et « Composant TextInput », à la page 1257.

Pour créer une case à cocher à l'aide d'ActionScript :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser le composant CheckBox du panneau Composants à la bibliothèque du document actuel.

   Cette opération permet d'ajouter le composant à la bibliothèque, mais elle ne permet pas de le rendre visible dans l'application.
3. Faites glisser le composant TextInput du panneau Composants à la bibliothèque du document actuel.
4. Dans la première image du scénario principal, ajoutez le code ActionScript suivant au panneau Actions pour créer des occurrences du composant et les positionner :

```
this.createClassObject(mx.controls.CheckBox, "testAge_ch", 1,
  {label:'Age Range', selected:true});
this.createClassObject(mx.controls.TextInput, "minimumAge_ti", 2,
  {restrict:[0-9], text:18, maxChars:2});
minimumAge_ti.move(20, 30);
this.createClassObject(mx.controls.TextInput, "maximumAge_ti", 3,
  {restrict:[0-9], text:55, maxChars:2});
maximumAge_ti.move(20, 60);
```

   La méthode « UIObject.createClassObject() », à la page 1416 est utilisée pour créer l'occurrence du composant CheckBox appelée **restrictAge**, et une propriété label est définie. Le code utilise ensuite la méthode « UIObject.move() », à la page 1429 pour positionner le bouton.
5. Ajoutez maintenant le code ActionScript suivant pour créer un écouteur d'événements et une fonction de gestionnaire d'événements :

```
// Création du gestionnaire pour l'événement checkBox.
function checkboxHandler(evt_obj:Object) {
 minimumAge_ti.enabled = evt_obj.target.selected;
 maximumAge_ti.enabled = evt_obj.target.selected;
}
// Ajout de l'écouteur.
testAge_ch.addEventListener("click", checkboxHandler);
```

   Ce code crée un gestionnaire d'événements `click` qui active et désactive les composants des champs de texte `minimumAge` et `maximumAge`. Pour plus d'informations, reportez-vous à `CheckBox.click`, à « EventDispatcher.addEventListener() », à la page 517 et à « Composant TextInput », à la page 1257.

Personnalisation du composant CheckBox

Vous pouvez orienter un composant CheckBox dans le sens horizontal et vertical pendant la création et à l'exécution. Lors de la programmation, sélectionnez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. Lors de l'exécution, utilisez la méthode `setSize()` (`UIObject.setSize()`) ou toutes les propriétés et méthodes applicables de Classe CheckBox. Le redimensionnement de la case à cocher ne modifie pas la taille de l'étiquette ni de l'icône, mais uniquement la taille du cadre de sélection.

Le cadre de sélection d'une occurrence de case est invisible et désigne également la zone active de l'occurrence. Si vous augmentez la taille de l'occurrence, vous augmentez également celle de la zone active. Si le cadre de sélection est trop petit pour contenir l'étiquette, celle-ci est découpée à la bonne taille.

Utilisation des styles avec le composant CheckBox

Vous pouvez définir des propriétés de style pour modifier l'aspect d'une occurrence CheckBox. Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, consultez « Utilisation de styles pour personnaliser la couleur et le texte des composants » dans *Utilisation des composants ActionScript 2.0*.

Un composant CheckBox prend en charge les styles suivants :

Style	Thème	Description
`themeColor`	Halo	Couleur de base d'un composant. `"haloGreen"`, `"haloBlue"` et `"haloOrange"` sont des valeurs possibles. La valeur par défaut est `"haloGreen"`.
`color`	Les deux	Couleur du texte. La valeur par défaut est 0x0B333C pour le thème Halo et vide pour le thème Sample.
`disabledColor`	Les deux	Couleur du texte lorsque le composant est désactivé. La couleur par défaut est 0x848384 (gris foncé).

Style	Thème	Description
`embedFonts`	Les deux	Valeur booléenne qui indique si la police spécifiée dans `fontFamily` est de type intégré. Ce style doit être défini sur `true` si `fontFamily` fait référence à une police intégrée. Dans le cas contraire, la police intégrée n'est pas utilisée. Si ce style est défini sur `true` et que `fontFamily` ne fait pas référence à une police intégrée, aucun texte n'est affiché. La valeur par défaut est `false`.
`fontFamily`	Les deux	Nom de la police du texte. La valeur par défaut est `"_sans"`.
`fontSize`	Les deux	Taille, en points, de la police. La valeur par défaut est 10.
`fontStyle`	Les deux	Style de police : `"normal"` ou `"italic"`. La valeur par défaut est `"normal"`.
`fontWeight`	Les deux	Epaisseur de la police : `"none"` ou `"bold"`. La valeur par défaut est `"none"`. Tous les composants peuvent également accepter la valeur `"normal"` au lieu de `"none"` pendant un appel à la méthode `setStyle()`, mais les appels suivants à `getStyle()` renvoient `"none"`.
`textDecoration`	Les deux	Décoration du texte : `"none"` ou `"underline"`. La valeur par défaut est `"none"`.
`symbolBackgroundColor`	Sample	Couleur d'arrière-plan de la case à cocher. La valeur par défaut est 0xFFFFFF (blanc).
`symbolBackgroundDisabledColor`	Sample	Couleur d'arrière-plan de la case à cocher lorsqu'elle est désactivée. La valeur par défaut est 0xEFEEEF (gris clair).
`symbolBackgroundPressedColor`	Sample	Couleur d'arrière-plan de la case à cocher lorsqu'elle est enfoncée. La valeur par défaut est 0xFFFFFF (blanc).
`symbolColor`	Sample	Couleur de la coche. La valeur par défaut est 0x000000 (noir).
`symbolDisabledColor`	Sample	Couleur de la coche désactivée. La valeur par défaut est 0x848384 (gris foncé).

Utilisation des enveloppes avec le composant CheckBox

Le composant CheckBox utilise les symboles de la Bibliothèque pour représenter les états des boutons. Pour appliquer une enveloppe au composant CheckBox lors de la programmation, modifiez les symboles dans le panneau Bibliothèque. Les enveloppes de composants CheckBox sont situées dans le dossier Flash UI Components 2/Themes/MMDefault/CheckBox Assets/states de la bibliothèque du fichier HaloTheme.fla ou SampleTheme.fla. Pour plus d'informations, consultez la section « A propos de l'application d'enveloppes aux composants » dans *Utilisation des composants*.

Les composants CheckBox utilisent les propriétés d'enveloppe suivantes. Seules les icônes, préférées aux enveloppes, sont affichées pour la case à cocher.

Propriété	Description
`falseUpIcon`	Etat non coché relevé (normal). La valeur par défaut est `CheckFalseUp`.
`falseDownIcon`	Etat non coché enfoncé. La valeur par défaut est `CheckFalseDown`.
`falseOverIcon`	Etat non coché survolé. La valeur par défaut est `CheckFalseOver`.
`falseDisabledIcon`	Etat non coché désactivé. La valeur par défaut est `CheckFalseDisabled`.
`trueUpIcon`	Etat coché basculé. La valeur par défaut est `CheckTrueUp`.
`trueDownIcon`	Etat coché enfoncé. La valeur par défaut est `CheckTrueDown`.
`trueOverIcon`	Etat coché survolé. La valeur par défaut est `CheckTrueOver`.
`trueDisabledIcon`	Etat coché désactivé. La valeur par défaut est `CheckTrueDisabled`.

Chacune de ces propriétés correspond à l'icône qui signale l'état de la case à cocher.
Le composant CheckBox ne présente pas de bordure ni d'arrière-plan.

Pour créer des symboles de clip pour les enveloppes de cases à cocher :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Choisissez Fichier > Importer > Ouvrir une bibliothèque externe et sélectionnez le fichier HaloTheme.fla.

   Ce fichier est stocké dans le dossier Configuration au niveau de l'application.
   Pour connaître son emplacement exact dans votre système d'exploitation, reportez-vous à « Présentation des thèmes » dans *Utilisation des composants ActionScript 2.0*.

3. Dans le panneau Bibliothèque du thème, choisissez Flash UI Components 2/Themes/MMDefault et faites glisser le dossier CheckBox Assets dans la bibliothèque de votre document.
4. Développez le dossier CheckBox Assets/States dans la bibliothèque de votre document.
5. Ouvrez les symboles que vous désirez personnaliser.

   Par exemple, ouvrez le symbole CheckFalseDisabled.
6. Personnalisez le symbole selon vos besoins.

   Par exemple, appliquez la couleur gris clair au carré blanc intérieur.
7. Répétez les étapes 5 et 6 pour tous les symboles devant être personnalisés.

   Par exemple, changez également la couleur pour la case intérieure du symbole CheckTrueDisabled.
8. Cliquez sur le bouton Précédent pour revenir au scénario principal.
9. Faites glisser un composant CheckBox sur la scène.

   Pour cet exemple, faites glisser deux occurrences afin d'illustrer les deux nouveaux symboles d'enveloppe.
10. Définissez les propriétés des occurrences CheckBox nécessaires.

    Pour cet exemple, définissez une occurrence CheckBox sur `true` et utilisez ActionScript pour définir les deux occurrences CheckBox sur désactivé.
11. Choisissez Contrôle > Tester l'animation.

Classe CheckBox

Héritage MovieClip > Classe UIObject > Classe UIComponent > Classe SimpleButton > Composant Button > CheckBox

Nom de classe ActionScript mx.controls.CheckBox

Les propriétés de la classe CheckBox permettent de créer une étiquette de texte et de la placer à gauche, à droite, au-dessus ou au-dessous d'une case à cocher lors de l'exécution.

La définition d'une propriété de la classe CheckBox avec ActionScript annule le paramètre du même nom défini dans l'inspecteur des propriétés ou des composants.

Le composant CheckBox utilise le gestionnaire de focus pour remplacer le rectangle de focus par défaut de Flash Player et tracer un rectangle de focus personnalisé aux coins arrondis. Pour plus d'informations, consultez « Création de la navigation personnalisée du focus » dans *Utilisation des composants ActionScript 2.0*.

Toutes les classes de composants ont une propriété `version` qui correspond à une propriété de classe. Les propriétés de classe ne sont disponibles que pour la classe elle-même. La propriété `version` renvoie une chaîne qui indique la version du composant. Pour accéder à cette propriété, utilisez le code suivant :

```
trace(mx.controls.CheckBox.version);
```

REMARQUE

`Le code trace(myCheckBoxInstance.version);` renvoie `undefined`.

Méthodes de la classe CheckBox

La classe CheckBox ne présente pas de méthodes exclusives.

Méthodes héritées de la classe UIObject

Le tableau suivant énumère les méthodes de la classe CheckBox héritées de la classe UIObject. Pour appeler ces méthodes à partir de l'objet CheckBox, utilisez le formulaire *`checkBoxInstance.methodName`*.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés ou des composants.
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
`UIObject.setSize()`	Redimensionne l'objet à la taille demandée.
`UIObject.setSkin()`	Définit une enveloppe dans l'objet.
`UIObject.setStyle()`	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant énumère les méthodes de la classe CheckBox héritées de la classe UIComponent. Pour appeler ces méthodes à partir de l'objet CheckBox, utilisez le formulaire *checkBoxInstance.methodName*.

Méthode	Description
`UIComponent.getFocus()`	Renvoie une référence à l'objet ayant le focus.
`UIComponent.setFocus()`	Attribue le focus à l'occurrence de composant.

Propriétés de la classe CheckBox

Le tableau suivant présente les propriétés de la classe CheckBox.

Propriété	Description
`CheckBox.label`	Spécifie le texte qui apparaît à côté d'une case à cocher.
`CheckBox.labelPlacement`	Spécifie l'orientation du texte de l'étiquette par rapport à la case.
`CheckBox.selected`	Spécifie si la case à cocher est activée (`true`) ou désactivée (`false`).

Propriétés héritées de la classe UIObject

Le tableau suivant énumère les propriétés de la classe CheckBox héritées de la classe UIObject. Lors de l'accès à ces propriétés à partir de l'objet CheckBox, utilisez le formulaire *checkBoxInstance.propertyName*.

Propriété	Description
`UIObject.bottom`	Lecture seule ; position du bord inférieur de l'objet par rapport au bord inférieur de son parent.
`UIObject.height`	Lecture seule ; hauteur de l'objet, en pixels.
`UIObject.left`	Lecture seule ; bord gauche de l'objet, en pixels.
`UIObject.right`	Lecture seule ; position du bord droit de l'objet par rapport au bord droit de son parent.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.

Propriété	Description
UIObject.top	Lecture seule ; renvoie la position du bord supérieur de l'objet par rapport à son parent.
UIObject.visible	Valeur booléenne indiquant si l'objet est visible (true) ou non (false).
UIObject.width	Lecture seule ; largeur de l'objet, en pixels.
UIObject.x	Lecture seule ; bord gauche de l'objet, en pixels.
UIObject.y	Lecture seule ; bord supérieur de l'objet, en pixels.

Propriétés héritées de la classe UIComponent

Le tableau suivant énumère les propriétés de la classe CheckBox héritées de la classe UIComponent. Lors de l'accès à ces propriétés à partir de l'objet CheckBox, utilisez le formulaire *checkBoxInstance.propertyName*.

Propriété	Description
UIComponent.enabled	Indique si le composant peut recevoir le focus et la saisie.
UIComponent.tabIndex	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Propriétés héritées de la classe SimpleButton

Le tableau suivant énumère les propriétés de la classe CheckBox héritées de la classe SimpleButton. Lors de l'accès à ces propriétés à partir de l'objet CheckBox, utilisez le formulaire *checkBoxInstance.propertyName*.

Propriété	Description
SimpleButton.emphasized	Indique si un bouton a l'aspect d'un bouton-poussoir par défaut.
SimpleButton.emphasizedStyleDeclaration	Déclaration de style lorsque la propriété emphasized est définie sur true.
SimpleButton.selected	Valeur booléenne indiquant si le bouton est sélectionné (true) ou non (false). La valeur par défaut est false.
SimpleButton.toggle	Valeur booléenne indiquant si le comportement du bouton est celui d'un bouton bascule (true) ou non (false). La valeur par défaut est false.

Propriétés héritées de la classe Button

Le tableau suivant énumère les propriétés de la classe CheckBox héritées de la classe Button. Lors de l'accès à ces propriétés à partir de l'objet CheckBox, utilisez le formulaire *checkBoxInstance.propertyName*.

Propriété	Description
`Button.label`	Spécifie le texte qui apparaît dans un bouton.
`Button.labelPlacement`	Spécifie l'orientation du texte de l'étiquette par rapport à une icône.

Evénements de la classe CheckBox

Le tableau suivant présente un événement de la classe CheckBox.

Evénement	Description
`CheckBox.click`	Déclenché lorsque le bouton de la souris est relâché sur la case à cocher ou si cette dernière a le focus et que la barre d'espace est enfoncée.

Evénements hérités de la classe UIObject

Le tableau suivant énumère les événements de la classe CheckBox hérités de la classe UIObject.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant énumère les événements de la classe CheckBox hérités de la classe UIComponent.

Evénement	Description
`UIComponent.focusIn`	Diffusé lorsqu'un objet reçoit le focus.
`UIComponent.focusOut`	Diffusé lorsqu'un objet perd le focus.
`UIComponent.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIComponent.keyUp`	Diffusé lorsqu'une touche est relâchée.

Evénements hérités de la classe SimpleButton

Le tableau suivant présente l'événement de la classe CheckBox hérité de la classe SimpleButton.

Evénement	Description
`SimpleButton.click`	Diffusé lorsque l'utilisateur clique sur un bouton.

CheckBox.click

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.click = function(eventObject:Object) {
  // ...
};
checkBoxInstance.addEventListener("click", listenerObject);
```

Utilisation 2 :

```
on (click) {
  // ...
}
```

Description

Evénement diffusé à tous les écouteurs enregistrés lorsque l'utilisateur clique puis relâche le bouton de la souris sur la case à cocher ou si cette dernière a le focus et que la barre d'espace est enfoncée.

Le premier exemple d'utilisation fait appel à un modèle dispatcher (diffuseur)/écouteur d'événement. Une occurrence de composant (*checkBoxInstance*) distribue un événement (ici, `click`) géré par une fonction (également appelée *gestionnaire*) sur un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée au déclenchement de l'événement. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. L'objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `addEventListener()` (voir `EventDispatcher.addEventListener()`) sur l'occurrence de composant qui diffuse l'événement afin d'enregistrer l'écouteur avec cette occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence CheckBox. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de composant. Par exemple, le code suivant, lié à la case à cocher `myCheckBox`, envoie « _level0.myCheckBox » au panneau Sortie :

```
on (click) {
  trace(this);
}
```

Exemple

L'exemple suivant active un bouton lorsque la case à cocher est sélectionnée. Cet exemple suppose qu'une occurrence du composant Button nommée `submit_button` et qu'une occurrence du composant CheckBox nommée `agree_ch` se trouvent sur la scène. Ajoutez le code suivant à la première image du scénario principal :

```
agree_ch.label = "I agree";
submit_button.enabled = false;

// Création d'un objet écouteur.
var form_obj:Object = new Object();

// Affectation d'une fonction à l'objet écouteur.
form_obj.click = function(event_obj:Object) {
 submit_button.enabled = event_obj.target.selected;
};

// Ajout de l'écouteur.
agree_ch.addEventListener("click", form_obj);
```

Le code suivant envoie un message au panneau Sortie lorsque l'utilisateur clique sur `checkBoxInstance`. Le gestionnaire `on()` doit être directement lié à `checkBoxInstance` :

```
on (click) {
  trace("check box component was clicked");
}
```

Voir aussi

`EventDispatcher.addEventListener()`

CheckBox.label

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`checkBoxInstance``.label`

Description

Propriété qui spécifie l'étiquette de texte de la case à cocher. Par défaut, l'étiquette apparaît à droite de la case à cocher. La définition de cette propriété remplace le paramètre d'étiquette spécifié dans l'onglet Paramètres de l'inspecteur des composants.

Le composant CheckBox interdit les étiquettes comportant plusieurs lignes.

Exemple

Le code suivant définit le texte qui apparaît en regard du composant CheckBox et envoie la valeur au panneau Sortie :

```
checkBox.label = "Remove from list";
trace(checkBox.label)
```

Cet exemple crée la case à cocher dans ActionScript puis redimensionne l'étiquette lorsqu'elle est sélectionnée. Pour cet exemple, faites glisser le composant CheckBox du panneau Composants jusqu'à la bibliothèque du document actuel (de façon à ce que le composant apparaisse dans la bibliothèque mais pas sur la scène). Ajoutez ensuite le code ActionScript suivant à la première image du scénario principal :

```
this.createClassObject(mx.controls.CheckBox, "my_ch", 10, {label:"Resize
  CheckBox instance"});

function checkboxHandler(evt_obj:Object):Void {
 trace("before: " + evt_obj.target.width +"px wide");
 evt_obj.target.setSize(200, evt_obj.target.height);
 trace("after: " + evt_obj.target.width+"px wide");
}
my_ch.addEventListener("click", checkboxHandler);
```

Voir aussi

`CheckBox.labelPlacement`

CheckBox.labelPlacement

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`checkBoxInstance``.labelPlacement`

Description

Propriété : chaîne indiquant la position de l'étiquette par rapport à la case à cocher. Les quatre valeurs possibles sont énumérées ci-dessous. Les pointillés représentent le cadre de délimitation du composant et sont invisibles dans un document.

- `"right"` La case à cocher est verrouillée dans le coin supérieur gauche du cadre de délimitation. L'étiquette est placée à sa droite. Il s'agit de la valeur par défaut.

- `"left"` La case à cocher est verrouillée dans le coin supérieur droit du cadre de délimitation. L'étiquette est placée à sa gauche.

- `"bottom"` L'étiquette est placée en dessous de la case à cocher. La case à cocher et l'étiquette sont centrées horizontalement et verticalement.

- `"top"` L'étiquette est placée au-dessus de la case à cocher. La case à cocher et l'étiquette sont centrées horizontalement et verticalement.

Vous pouvez modifier le cadre de délimitation d'un composant au cours de la création à l'aide de la commande Transformer ou pendant l'exécution au moyen de la propriété `UIObject.setSize()`. Pour plus d'informations, voir « Personnalisation du composant CheckBox », à la page 137.

Exemple

Dans l'exemple suivant, l'étiquette est placée à gauche de la case à cocher :

```
checkBox_mc.labelPlacement = "left";
```

L'exemple suivant utilise ActionScript pour créer des occurrences de la case à cocher. Les propriétés label et labelPlacement de l'occurrence CheckBox `right_ch` sont définies dans la méthode « UIObject.createClassObject() », à la page 1416. Les propriétés label et labelPlacement de l'occurrence CheckBox `left_ch` sont définies dans des déclarations distinctes. Faites glisser le composant CheckBox du panneau Composants à la bibliothèque du document en cours (pour que le composant apparaisse dans la bibliothèque, mais pas sur la scène). Ajoutez ensuite le code ActionScript suivant à la première image du scénario principal :

```
this.createClassObject(mx.controls.CheckBox, "right_ch", 1, {label:"Right",
  labelPlacement:"right"});
right_ch.move(10, 10);
this.createClassObject(mx.controls.CheckBox, "left_ch", 2);
left_ch.label= "Left";
left_ch.labelPlacement = "left";
left_ch.move(10, 30);
```

Voir aussi

`CheckBox.label`

CheckBox.selected

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

checkBoxInstance.selected

Description

Propriété : valeur booléenne qui active (`true`) ou désactive (`false`) la case à cocher.

Exemple

L'exemple suivant illustre une case à cocher dont la propriété `selected` est définie sur `true`, par défaut, puis utilise la propriété `selected` dans une fonction de gestionnaire d'événements pour répondre à l'utilisateur qui clique sur la case à cocher. Faites glisser un composant CheckBox sur la scène. Entrez le nom d'occurrence `my_ch`. Ajoutez ensuite le code suivant au panneau Actions de la première image du scénario principal :

```
my_ch.selected = true;

var checkboxListener:Object = new Object();
checkboxListener.click = function(evt_obj:Object) {
 if (evt_obj.target.selected) {
  evt_obj.target.label = "Selected!";
 } else {
  evt_obj.target.label = "Unselected!";
 }
};
my_ch.addEventListener("click", checkboxListener);
```

CHAPITRE 7

Interface Collection

7

La classe Collection est distribuée dans la bibliothèque des classes communes sous forme de symbole de clip compilé. Pour accéder à cette classe, choisissez Fenêtre > Bibliothèques communes > Classes, qui contient le clip compilé UtilsClasses.

Classe Collection

Nom de classe ActionScript mx.utils.Collection

L'interface Collection permet de gérer par programmation un groupe d'éléments apparentés, appelés *éléments de collection*. Chaque élément de collection de cet ensemble présente des propriétés décrites dans les métadonnées de sa définition.

REMARQUE	La classe Collection est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Les composants peuvent exposer leurs propriétés sous forme de collections, que vous pouvez manipuler lors de la programmation via la boîte de dialogue Valeurs de l'inspecteur des composants. Cette boîte de dialogue vous permet d'ajouter des éléments, d'en supprimer, de modifier leurs propriétés et leur position au sein de la collection. Pour plus d'informations sur les collections et les éléments de collection, reportez-vous à « Présentation de la balise Collection » dans *Utilisation des composants ActionScript 2.0*.

L'interface Collection est généralement utilisée avec les composants qui créent des propriétés de collection à l'aide de la balise de métadonnées Collection. Même si vous pouvez par programmation créer des occurrences de collection, accéder à ces occurrences et les supprimer, les collections sont plus fréquemment utilisées dans le contexte d'un composant. Flash fournit les implémentations des deux interfaces qui sont liées aux collections (CollectionImpl pour Collection et IteratorImpl pour Iterator).

Récapitulatif des méthodes de l'interface Collection

Le tableau suivant présente les méthodes de l'interface Collection.

Méthode	Description
`Collection.addItem()`	Ajoute un nouvel élément à la fin de la collection.
`Collection.contains()`	Indique si la collection contient l'élément spécifié.
`Collection.clear()`	Supprime tous les éléments de la collection.
`Collection.getItemAt()`	Renvoie un élément de la collection à l'aide de son index.
`Collection.getIterator()`	Renvoie un itérateur sur les éléments de la collection.
`Collection.getLength()`	Renvoie le nombre d'éléments de la collection.
`Collection.isEmpty()`	Indique si la collection est vide.
`Collection.removeItem()`	Supprime l'élément spécifié de la collection.

Collection.addItem()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`collection``.addItem(`*`item`*`)`

Paramètres

`item` Objet à ajouter à la collection. Si *`item`* est `null`, l'élément n'est pas ajouté à la collection.

Renvoie

Valeur booléenne `true` si la collection a été modifiée par l'opération.

Description

Méthode qui ajoute un nouvel élément à la fin de la collection.

Exemple

L'exemple suivant appelle `addItem()` :

```
on (click) {
import CompactDisc;

  var myColl:mx.utils.Collection;
  myColl = _parent.thisShelf.MyCompactDiscs;
  myCD = new CompactDisc();
  myCD.Artist = "John Coltrane";
  myCD.Title = "Giant Steps";

  var wasAdded:Boolean = myColl.addItem(myCD);
}
```

Collection.contains()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`collection``.contains(`*`item`*`)`

Paramètres

`item` Objet dont la présence dans la collection doit être testée.

Renvoie

Valeur booléenne `true` si la collection contient *`item`*.

Description

Méthode qui indique si la collection contient l'élément spécifié. Pour que Flash considère ces objets comme égaux, ils doivent faire référence au même objet. Si *`item`* est un objet différent, `Collection.contains()` renvoie `false`, même si les propriétés de l'objet sont toutes égales.

Exemple

L'exemple suivant appelle `contains()` :

```
var myColl:mx.utils.Collection;
myColl = _parent.thisShelf.MyCompactDiscs;

var itr:mx.utils.Iterator = myColl.getIterator();
while (itr.hasNext()) {
  var cd:CompactDisc = CompactDisc(itr.next());
  var title:String = cd.Title;
  var artist:String = cd.Artist;

  if(myColl.contains(cd)) {
    trace("myColl contains " + title);
  }
  else {
    trace("myColl does not contain " + title);
  }
}
```

Collection.clear()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

collection`.clear()`

Renvoie

Aucune.

Description

Méthode qui supprime tous les éléments de la collection.

Exemple

L'exemple suivant appelle `clear()` :

```
on (click) {
  var myColl:mx.utils.Collection;
  myColl = _parent.thisShelf.MyCompactDiscs;
  myColl.clear();
}
```

Collection.getItemAt()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

collection.getItemAt(*index*)

Paramètres

index Nombre indiquant la position de *item* au sein de la collection. Ce numéro d'index est basé sur zéro : 0 récupère le premier élément, 1 le second, et ainsi de suite.

Renvoie

Objet contenant une référence à l'élément de collection spécifié ou `null` si *index* sort des limites.

Description

Méthode qui renvoie un élément de la collection à l'aide de son index.

Exemple

L'exemple suivant appelle `getItemAt()` :

```
//...
  var myColl:mx.utils.Collection;
  myColl = _parent.thisShelf.MyCompactDiscs;
  var myCD = CompactDisc(myColl.getItemAt(0));
  if (myCD !=null) {
    trace("Retrieved " + myCD.Title);
  }
//...
```

Collection.getIterator()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

collection.getIterator()

Renvoie

Objet Iterator qui vous permet de parcourir la collection.

Description

Méthode qui renvoie un itérateur sur les éléments de la collection. L'ordre dans lequel les éléments sont renvoyés n'est pas garanti (sauf si cette collection est une occurrence d'une classe qui offre une garantie).

Exemple

L'exemple suivant appelle `getIterator()` :

```
on (click) {
  var myColl:mx.utils.Collection;
  myColl = _parent.thisShelf.MyCompactDiscs;

  var itr:mx.utils.Iterator = myColl.getIterator();
  while (itr.hasNext()) {
    var cd:CompactDisk = CompactDisc(itr.next());
    var title:String = cd.Title;
    var artist:String = cd.Artist;

    trace("Title: " + title + " - Artist: " + artist);
  }
}
```

Collection.getLength()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

collection.getLength()

Renvoie

Nombre d'éléments de la collection.

Description

Méthode qui renvoie le nombre d'éléments de la collection.

Exemple

L'exemple suivant appelle `getLength()` :

```
//...
var myColl:mx.utils.Collection;
myColl = _parent.thisShelf.MyCompactDiscs;
trace ("Collection size is: " + myColl.getLength());
//...
```

Collection.isEmpty()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

collection.isEmpty()

Renvoie

Valeur booléenne `true` si la collection est vide.

Description

Méthode qui indique si la collection est vide.

Exemple

L'exemple suivant appelle `isEmpty()` :

```
on (click) {
  var myColl:mx.utils.Collection;
  myColl = _parent.thisShelf.MyCompactDiscs;
  if (myColl.isEmpty()) {
    trace("No CDs in the collection");
  }
}
//...
```

Collection.removeItem()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`collection``.removeItem(`*`item`*`)`

Paramètres

`item` Objet à supprimer de la collection.

Renvoie

Valeur booléenne `true` si *`item`* a bien été supprimé.

Description

Méthode qui supprime l'élément spécifié de la collection. `Collection.removeItem()` réduisant de façon dynamique la taille de la collection, n'appelez pas cette méthode pendant une boucle effectuée par l'intermédiaire d'un itérateur.

Exemple

L'exemple suivant appelle `removeItem()` :

```
var myColl:mx.utils.Collection;
myColl = _parent.thisShelf.MyCompactDiscs;

// Obtention à partir d'un champ de saisie de texte.
var removeArtist:String = _parent.tArtistToRemove.text;
var removeSize:Number = 0;

if (myColl.isEmpty()) {
  trace("No CDs in the collection");
}
else {
  var toRemove:Array = new Array();
  var itr:mx.utils.Iterator = myColl.getIterator();
  var cd:CompactDisc = new CompactDisc();
  var title:String = "";
  var artist:String = "";
  while (itr.hasNext()) {
    cd = CompactDisc(itr.next());
    title = cd.Title;
    artist = cd.Artist;
    if(artist == removeArtist) {
      // Marquage de cette artiste pour la suppression.
      removeSize = toRemove.push(cd);
      trace("*** Marked for deletion: " + artist + "|" + title);
    }
  }
  // Après une boucle, suppression des éléments incorrects.
  var removeCD:CompactDisc = new CompactDisc();
  for(i = 0; i < removeSize; i++) {
    removeCD = toRemove[i];
    trace("Removing: " + removeCD.Artist + "|" + removeCD.Title);
    myColl.removeItem(removeCD);
  }
}
```

CHAPITRE 8

Composant ComboBox

8

Une liste déroulante permet à un utilisateur d'effectuer une sélection unique. Cette liste déroulante peut être statique ou modifiable. Si elle est modifiable, elle permet à l'utilisateur de saisir du texte directement dans un champ de texte en haut de la liste et de sélectionner un élément. Si la liste déroulante atteint le bas du document, elle se déroule vers le haut et non vers le bas. Elle comporte trois sous-composants : Button, TextInput et List.

REMARQUE

Un composant ComboBox est pris en charge pour ActionScript 2.0 et ActionScript 3.0. Ce document présente le composant de la version 2. Si vous utilisez celui de la version 3, reportez-vous à « Utilisation du composant ComboBox » dans *Utilisation des composants ActionScript 3.0*.

Lorsqu'un élément est sélectionné dans la liste déroulante, son étiquette est copiée dans le champ de texte, en haut de la liste. La méthode utilisée pour effectuer la sélection, souris ou clavier, importe peu.

Un composant ComboBox reçoit le focus lorsque vous cliquez sur son champ de texte ou sur son bouton. Lorsqu'un composant ComboBox a le focus et peut être modifié, toutes les frappes de touches vont dans la zone de texte et sont traitées selon les règles du composant TextInput (voir « Composant TextInput », à la page 1257), à l'exception des touches suivantes :

Touche	Description
Ctrl+Flèche vers le bas	Ouvre la liste déroulante et lui attribue le focus.
Maj+Tab	Place le focus sur l'objet précédent.
Tab	Place le focus sur l'objet suivant.

Lorsqu'un composant ComboBox statique a le focus, les frappes sur les touches alphanumériques déplacent la sélection vers le haut et le bas de la liste déroulante pour atteindre l'élément suivant commençant par le même caractère. Vous pouvez également utiliser les touches suivantes pour contrôler une liste déroulante statique :

Touche	Description
Ctrl+Flèche vers le bas	Ouvre la liste déroulante et lui attribue le focus.
Ctrl+Flèche vers le haut	Ferme la liste déroulante, si elle est ouverte dans les versions autonome et navigateur de Flash Player.
Flèche vers le bas	Déplace la sélection d'un élément vers le bas.
Fin	La sélection se déplace jusqu'au bout de la liste.
Echap	Ferme la liste déroulante et renvoie le focus sur la liste déroulante en mode test.
Entrée	Ferme la liste déroulante et place à nouveau le focus sur le composant ComboBox.
Page d'accueil	Déplace la sélection en haut de la liste.
Pg. Suiv.	Déplace la sélection d'une page vers le bas.
Pg. Préc.	Déplace la sélection d'une page vers le haut.
Maj+Tab	Place le focus sur l'objet précédent.
Tab	Place le focus sur l'objet suivant.

Lorsque la liste déroulante d'un composant ComboBox a le focus, les frappes sur les touches alphanumériques déplacent la sélection vers le haut et le bas de la liste pour atteindre l'élément suivant commençant par le même caractère. Vous pouvez également utiliser les touches suivantes pour contrôler une liste déroulante :

Touche	Description
Ctrl+Flèche vers le haut	Si la liste déroulante est ouverte, le focus retourne au champ de texte et la liste déroulante se ferme dans les versions autonome et navigateur de Flash Player.
Flèche vers le bas	Déplace la sélection d'un élément vers le bas.
Fin	Place le point d'insertion à la fin du champ de texte.
Entrée	Si la liste déroulante est ouverte, le focus retourne au champ de texte et la liste se ferme.

Touche	Description
Echap	Si la liste déroulante est ouverte, le focus retourne dans la zone de texte et la liste déroulante est fermée en mode test.
Origine	Place le point d'insertion au début du champ de texte.
Page suivante	Déplace la sélection d'une page vers le bas.
Page précédente	Déplace la sélection d'une page vers le haut.
Tab	Place le focus sur l'objet suivant.
Maj+Fin	Sélectionne le texte compris entre le point d'insertion et la fin du champ de texte.
Maj+Origine	Sélectionne le texte compris entre le point d'insertion et le début du champ de texte.
Maj+Tab	Place le focus sur l'objet précédent.
Flèche vers le haut	Déplace la sélection d'un élément vers le haut.

REMARQUE

La taille de page utilisée par les touches Page précédente et Page suivante correspond au nombre d'éléments contenus dans l'affichage, moins un. Par exemple, le passage à la page suivante dans une liste déroulante à dix lignes affichera les éléments 0-9, 9-18, 18-27, etc., avec un élément commun par page.

Pour plus d'informations sur le contrôle du focus, reportez-vous à « Classe FocusManager », à la page 745 ou à « Création de la navigation personnalisée du focus » dans *Utilisation des composants ActionScript 2.0*.

L'aperçu en direct de chaque occurrence de composant ComboBox sur la scène reflète les modifications apportées aux paramètres dans l'inspecteur des propriétés ou des composants au cours de la programmation. Cependant, la liste déroulante ne s'ouvre pas en aperçu en direct et le premier élément apparaît comme étant l'élément sélectionné.

Lorsque vous ajoutez le composant ComboBox à une application, vous pouvez utiliser le panneau Accessibilité pour le rendre accessible aux lecteurs d'écran. Commencez par ajouter la ligne de code suivante pour activer l'accessibilité :

```
mx.accessibility.ComboBoxAccImpl.enableAccessibility();
```

Quel que soit le nombre d'occurrences d'un composant, l'activation de son accessibilité ne se fait qu'une fois. Pour plus d'informations, reportez-vous à *Création de contenu accessible* dans *Utilisation de Flash*.

Utilisation du composant ComboBox

Vous pouvez utiliser un composant ComboBox dans tout formulaire ou toute application qui requiert un choix unique dans une liste. Par exemple, vous pouvez fournir une liste déroulante de pays dans un formulaire où les clients doivent saisir leur adresse. Les listes déroulantes modifiables conviennent pour des scénarios plus complexes. Par exemple, dans une application d'itinéraire routier, utilisez une liste déroulante modifiable pour que l'utilisateur y saisisse ses adresses de départ et d'arrivée. La liste déroulante pourrait alors contenir des adresses déjà saisies.

Paramètres du composant ComboBox

Dans l'inspecteur des propriétés ou l'inspecteur de composants (option de menu Fenêtre > Inspecteur de composants), vous pouvez définir les paramètres de création suivants pour chaque occurrence ComboBox :

data associe une valeur de données avec chaque élément du composant ComboBox. Le paramètre data est un tableau.

editable détermine si le composant ComboBox est modifiable (`true`) ou uniquement sélectionnable (`false`). La valeur par défaut est `false`.

labels remplit le composant ComboBox avec un tableau de valeurs de texte.

rowCount définit le nombre maximal d'éléments affichables dans une liste. La valeur par défaut est 5.

Dans l'inspecteur des composants, vous pouvez définir les paramètres supplémentaires suivants pour chaque occurrence du composant ComboBox (Fenêtre > Inspecteur de composants) :

restrict désigne le jeu de caractères qu'un utilisateur peut saisir dans le champ de texte d'une liste déroulante. La valeur par défaut est `undefined`. Voir « ComboBox.restrict », à la page 200.

enabled est une valeur booléenne qui indique si le composant peut recevoir le focus et l'entrée. La valeur par défaut est `true`.

visible est une valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`). La valeur par défaut est `true`.

REMARQUE

Les propriétés `minHeight` et `minWidth` sont utilisées par des routines de dimensionnement internes. Elles sont définies dans UIObject et remplacées par différents composants selon vos besoins. Ces propriétés peuvent être utilisées pour créer un gestionnaire de dispositions personnalisées. Dans le cas contraire, la définition de ces propriétés dans l'inspecteur de composants n'a aucun effet visible.

Vous pouvez rédiger du code ActionScript pour définir d'autres options pour les occurrences ComboBox à l'aide des méthodes, propriétés et événements de la classe ComboBox. Pour plus d'informations, voir « Classe ComboBox », à la page 171.

Création d'une application avec le composant ComboBox

La procédure suivante décrit l'ajout d'un composant ComboBox à une application au cours de la programmation. Dans cet exemple, la liste déroulante présente des villes à sélectionner.

Pour créer une application avec le composant ComboBox :

1. Sélectionnez Fichier > Nouveau, puis choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser un composant ComboBox du panneau Composants jusqu'à la scène.
3. Sélectionnez l'outil Transformer et redimensionnez le composant sur la scène.

   La liste déroulante ne peut être redimensionnée sur la scène qu'au cours de la programmation. En général, on ne modifie que sa largeur pour que les entrées y soient correctement affichées.
4. Sélectionnez la liste déroulante, puis, dans l'inspecteur Propriétés, saisissez le nom d'occurrence **comboBox**.
5. Dans l'inspecteur des propriétés ou des composants, procédez comme suit :
   - Saisissez **Minneapolis**, **Portland** et **Keene** comme paramètre d'étiquette. Double-cliquez sur le champ du paramètre de l'étiquette pour ouvrir la boîte de dialogue Valeurs. Cliquez ensuite sur le signe plus pour ajouter des éléments.
   - Saisissez **MN.swf**, **OR.swf** et **NH.swf** comme paramètre de données.

     Il s'agit de fichiers SWF fictifs qui peuvent par exemple être chargés lorsqu'un utilisateur sélectionne une ville dans la liste déroulante.
6. Sélectionnez l'image 1 dans le scénario, ouvrez le panneau Actions et saisissez le code suivant :

```
function change(evt){
  trace(evt.target.selectedItem.label);
}
comboBox.addEventListener("change", this);
```

   La dernière ligne de code ajoute un gestionnaire d'événements `change` à l'occurrence `ComboBox`. Pour plus d'informations, reportez-vous à `ComboBox.change`.

Pour créer un composant ComboBox à l'aide d'ActionScript :

1. Sélectionnez Fichier > Nouveau, puis choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser le composant ComboBox du panneau Composants jusqu'à la bibliothèque du document actuel.

   Cette opération permet d'ajouter le composant à la bibliothèque, mais elle ne permet pas de le rendre visible dans l'application.
3. Sélectionnez la première image dans le scénario principal, ouvrez le panneau Actions et indiquez le code suivant :

```
this.createClassObject(mx.controls.ComboBox, "my_cb", 10);

my_cb.addItem({data:1, label:"One"});
my_cb.addItem({data:2, label:"Two"});
```

   Ce script utilise la méthode « UIObject.createClassObject() », à la page 1416 pour créer l'occurrence ComboBox, puis « ComboBox.addItem() », à la page 176 pour ajouter des éléments de la liste au composant ComboBox.
4. Ajoutez maintenant un écouteur d'événement et une fonction de gestionnaire d'événements pour répondre lorsqu'un élément ComboBox est sélectionné :

```
// Création d'un objet écouteur.
var cbListener:Object = new Object();
// Création d'une fonction de gestionnaire d'événements.
cbListener.change = function (evt_obj:Object) {
 trace("Currently selected item is: " +
   evt_obj.target.selectedItem.label);
}
// Ajout de l'écouteur d'événements.
my_cb.addEventListener("change", cbListener);
```

5. Choisissez Contrôle > Tester l'animation, puis cliquez sur un élément dans la liste déroulante pour afficher un message dans le panneau Sortie.

Personnalisation du composant ComboBox

Vous pouvez transformer un composant ComboBox horizontalement et verticalement au cours de la programmation. Lors de la programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation.

Si le texte est trop long pour tenir dans la liste déroulante, il est tronqué. Vous devez redimensionner la liste déroulante au cours de la programmation pour que le texte de l'étiquette tienne dans l'espace fourni.

Dans les listes déroulantes modifiables, le bouton est la seule zone active, pas le champ de texte. Pour les listes déroulantes statiques, le bouton et le champ de texte constituent la zone active. La zone active répond par l'ouverture ou la fermeture de la liste déroulante.

Utilisation de styles avec le composant ComboBox

Vous pouvez définir des propriétés de style pour modifier l'aspect d'un composant ComboBox. Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, reportez-vous à « Utilisation de styles pour personnaliser la couleur et le texte des composants » dans *Utilisation des composants ActionScript 2.0*.

La liste déroulante a deux styles uniques : `openDuration` et `openEasing`. Les autres styles sont transmis au bouton, au champ de texte et à la liste par le biais de ces composants individuels, de la manière suivante :

- Le bouton est une occurrence Button et en utilise les styles. (Voir la section « Utilisation de styles avec le composant Button », à la page 95).
- Le texte est une occurrence TextInput et en utilise les styles. (Voir la section « Utilisation de styles avec le composant TextInput », à la page 1261).
- La liste est une occurrence List et en utilise les styles. (Voir la section « Utilisation des styles avec le composant List », à la page 797).

Les composants ComboBox utilisent les styles suivants :

Style	Thème	Description
`themeColor`	Halo	Couleur de base d'un composant. `"haloGreen"`, `"haloBlue"` et `"haloOrange"` sont des valeurs possibles. La valeur par défaut est `"haloGreen"`.
`backgroundColor`	Les deux	Couleur d'arrière-plan. La couleur par défaut est le blanc.

Style	Thème	Description
`borderStyle`	Les deux	Le sous-composant Button utilise deux occurrences RectBorder comme bordures et répond au styles définis pour cette classe. Voir « Classe RectBorder », à la page 1111. Dans le thème Halo, le composant ComboBox utilise une bordure arrondie personnalisée pour la partie réduite de la liste déroulante. Les couleurs de cette partie du composant ComboBox sont modifiables via l'application d'enveloppes uniquement. Voir « Utilisation d'enveloppes avec le composant ComboBox », à la page 169.
`color`	Les deux	Couleur du texte. La valeur par défaut est 0x0B333C pour le thème Halo et vide pour le thème Sample.
`disabledColor`	Les deux	Couleur du texte lorsque le composant est désactivé. La couleur par défaut est 0x848384 (gris foncé).
`embedFonts`	Les deux	Valeur booléenne qui indique si la police spécifiée dans `fontFamily` est de type intégré. Ce style doit être défini sur `true` si `fontFamily` fait référence à une police intégrée. Dans le cas contraire, la police intégrée n'est pas utilisée. Si ce style est défini sur `true` et que `fontFamily` ne fait pas référence à une police intégrée, aucun texte n'est affiché. La valeur par défaut est `false`.
`fontFamily`	Les deux	Nom de la police du texte. La valeur par défaut est `"_sans"`.
`fontSize`	Les deux	Taille, en points, de la police. La valeur par défaut est 10.
`fontStyle`	Les deux	Style de police : `"normal"` ou `"italic"`. La valeur par défaut est `"normal"`.
`fontWeight`	Les deux	Epaisseur de la police : `"none"` ou `"bold"`. La valeur par défaut est `"none"`. Tous les composants peuvent également accepter la valeur `« normal »` au lieu de `« none »` pendant un appel de `setStyle()`, mais les prochains appels de `getStyle()` renvoient `« none »`.
`textAlign`	Les deux	Alignement du texte : « `left` », « `right` » ou « `center` ». La valeur par défaut est « `left` ».
`textDecoration`	Les deux	Décoration du texte : `"none"` ou `"underline"`. La valeur par défaut est `"none"`.

Style	Thème	Description
openDuration	Les deux	Durée, en millisecondes, de l'animation de transition. La valeur par défaut est 250.
openEasing	Les deux	Référence à une fonction d'interpolation qui contrôle l'animation. Par défaut, sine in/out (sinus). Pour plus d'informations, reportez-vous à « Personnalisation des animations de composants » dans *Utilisation des composants ActionScript 2.0*.

L'exemple suivant montre comment utiliser des styles List pour contrôler le comportement de la partie déroulante d'un composant ComboBox.

```
// comboBox est une occurrence du composant ComboBox sur la scène.
comboBox.setStyle("alternatingRowColors", [0xFFFFFF, 0xBFBFBF]);
```

Utilisation d'enveloppes avec le composant ComboBox

Le composant ComboBox utilise les symboles de la bibliothèque pour représenter les états des boutons et dispose de variables d'enveloppe pour la flèche vers le bas. Ces enveloppes sont situées dans le dossier Flash UI Components 2/Themes/MMDefault/ComboBox Assets/States des fichiers HaloTheme.fla et SampleTheme.fla. Les informations ci-dessous décrivent ces enveloppes et la procédure qui permet de les personnaliser.

Le composant ComboBox utilise également des enveloppes spéciales pour la barre de défilement de la liste déroulante et deux occurrences de la classe RectBorder pour la bordure qui encadre la zone de saisie de texte et la liste déroulante. Pour plus d'informations sur la personnalisation des enveloppes, reportez-vous à « Utilisation d'enveloppes avec le composant UIScrollBar », à la page 1448 et à « Classe RectBorder », à la page 1111. Pour plus d'informations sur les méthodes permettant d'appliquer des enveloppes aux composants, reportez-vous à « A propos de l'application d'enveloppes aux composants » dans *Utilisation des composants ActionScript 2.0*.

Les composants ComboBox utilisent les propriétés d'enveloppe suivantes :

Propriété	Description
ComboDownArrowDisabledName	Etat désactivé de la flèche bas. La valeur par défaut est ComboDownArrowDisabled.
ComboDownArrowDownName	Etat Enfoncé de la flèche bas. La valeur par défaut est ComboDownArrowDown.

Propriété	Description
`ComboDownArrowUpName`	Etat Relevé de la flèche bas. La valeur par défaut est `ComboDownArrowOver`.
`ComboDownArrowOverName`	Etat Survolé de la flèche bas. La valeur par défaut est `ComboDownArrowUp`.

Pour créer des symboles de clip pour les enveloppes ComboBox :

1. Sélectionnez Fichier > Nouveau, puis choisissez Fichier Flash (ActionScript 2.0).
2. Choisissez Fichier > Importer > Ouvrir une bibliothèque externe et sélectionnez le fichier HaloTheme.fla.

   Ce fichier est stocké dans le dossier Configuration au niveau de l'application. Pour connaître son emplacement exact dans votre système d'exploitation, reportez-vous à « Présentation des thèmes » dans *Utilisation des composants ActionScript 2.0*.
3. Dans le panneau Bibliothèque du thème, développez Flash UI Components 2/Themes/MMDefault et faites glisser le dossier ComboBox Assets vers la bibliothèque de votre document.
4. Développez le dossier ComboBox Assets/States dans la bibliothèque de votre document.
5. Ouvrez les symboles que vous désirez personnaliser.

   Par exemple, ouvrez le symbole ComboDownArrowDisabled.
6. Personnalisez le symbole selon vos besoins.

   Par exemple, appliquez la couleur gris clair au carré blanc intérieur.
7. Répétez les étapes 5 et 6 pour tous les symboles devant être personnalisés.
8. Cliquez sur le bouton Précédent pour revenir au scénario principal.
9. Faites glisser un composant ComboBox sur la scène.
10. Définissez les propriétés des occurrences ComboBox nécessaires.

    Pour cet exemple, utilisez ActionScript pour définir le composant ComboBox sur désactivé.
11. Choisissez Contrôle > Tester l'animation.

Classe ComboBox

Héritage MovieClip > Classe UIObject > Classe UIComponent > ComboBase > ComboBox

Nom de classe ActionScript mx.controls.ComboBox

Le composant ComboBox associe trois sous-composants distincts : Button, TextInput et List. La plupart des méthodes, propriétés et événements de chaque sous-composant sont disponibles directement depuis le composant ComboBox et énumérés dans les tableaux de la classe ComboBox.

La liste déroulante du composant ComboBox prend la forme d'un tableau ou d'un fournisseur de données. Si vous utilisez un fournisseur de données, la liste change lors de l'exécution. Vous pouvez modifier la source des données du composant ComboBox dynamiquement en basculant sur un nouveau tableau ou fournisseur de données.

Les éléments d'une liste déroulante sont indexés par position, en commençant par le chiffre 0. Les éléments peuvent être les suivants :

- Un type de données de base.
- Un objet contenant une propriété `label` et une propriété `data`.

REMARQUE

Un objet peut utiliser la propriété `ComboBox.labelFunction` ou `ComboBox.labelField` pour déterminer la propriété `label`.

Si le type de données de base de l'élément n'est pas une chaîne, il est converti en chaîne. Si l'élément est un objet, la propriété `label` doit être une chaîne, et la propriété `data` peut avoir n'importe quelle valeur ActionScript.

Les méthodes du composant ComboBox auxquelles vous fournissez des éléments ont deux paramètres, *`label`* et *`data`*, qui se réfèrent aux propriétés ci-dessus. Les méthodes qui renvoient un élément le renvoient en tant qu'objet.

Le composant ComboBox reporte l'instanciation de sa liste déroulante jusqu'à la prochaine interaction d'un utilisateur. De ce fait, la liste déroulante peut sembler lente à la première utilisation.

Pour accéder à la liste déroulante du composant ComboBox par programmation et annuler ce retard, servez-vous du code suivant :

```
var foo = myComboBox.dropdown;
```

L'accès à la liste déroulante peut entraîner une pause dans l'application. lorsque l'utilisateur utilise la liste déroulante pour la première fois ou lorsque le code ci-dessus s'exécute.

Méthodes de la classe ComboBox

Le tableau suivant présente les méthodes de la classe ComboBox.

Méthode	Description
`ComboBox.addItem()`	Ajoute un élément à la fin de la liste.
`ComboBox.addItemAt()`	Ajoute un élément à l'emplacement d'index spécifié.
`ComboBox.close()`	Ferme la liste déroulante.
`ComboBox.getItemAt()`	Renvoie l'élément à l'emplacement d'index spécifié.
`ComboBox.open()`	Ouvre la liste déroulante.
`ComboBox.removeAll()`	Supprime tous les éléments de la liste.
`ComboBox.removeItemAt()`	Supprime un élément de la liste à l'emplacement spécifié.
`ComboBox.replaceItemAt()`	Remplace le contenu de l'élément à l'index spécifié.
`ComboBox.sortItems()`	Trie la liste à l'aide d'une fonction de comparaison.
`ComboBox.sortItemsBy()`	Trie la liste à l'aide d'un champ de chaque élément.

Méthodes héritées de la classe UIObject

Le tableau suivant énumère les méthodes de la classe ComboBox héritées de la classe UIObject. Pour appeler ces méthodes à partir de l'objet ComboBox, utilisez le formulaire *`comboBoxInstance.methodName`*.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
`UIObject.setSize()`	Redimensionne l'objet à la taille demandée.

Méthode	Description
UIObject.setSkin()	Définit une enveloppe dans l'objet.
UIObject.setStyle()	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant énumère les méthodes de la classe ComboBox héritées de la classe UIComponent. Pour appeler ces méthodes à partir de l'objet ComboBox, utilisez le formulaire *comboBoxInstance.methodName*.

Méthode	Description
UIComponent.getFocus()	Renvoie une référence à l'objet ayant le focus.
UIComponent.setFocus()	Attribue le focus à l'occurrence de composant.

Propriétés de la classe ComboBox

Le tableau suivant présente les propriétés de la classe ComboBox.

Propriété	Description
ComboBox.dataProvider	Modèle de données pour les éléments de la liste.
ComboBox.dropdown	Renvoie une référence au composant List contenu dans ComboBox.
ComboBox.dropdownWidth	Largeur de la liste déroulante, en pixels.
ComboBox.editable	Indique si le composant ComboBox est modifiable.
ComboBox.labelField	Indique le champ de données à utiliser en tant qu'étiquette pour la liste déroulante.
ComboBox.labelFunction	Spécifie la fonction de calcul du champ de l'étiquette pour la liste déroulante.
ComboBox.length	Lecture seule : longueur de la liste déroulante.
ComboBox.restrict	Jeu de caractères qu'un utilisateur peut saisir dans le champ de texte d'une liste déroulante.
ComboBox.rowCount	Nombre maximal d'éléments de la liste à afficher en même temps.
ComboBox.selectedIndex	Index de l'élément sélectionné dans la liste déroulante.
ComboBox.selectedItem	Valeur de l'élément sélectionné dans la liste déroulante.

Propriété	Description
`ComboBox.text`	Chaîne de texte dans la zone de texte.
`ComboBox.textField`	Référence au composant TextInput dans la liste déroulante.
`ComboBox.value`	Valeur du champ de texte (modifiable) ou de la liste déroulante (statique).

Propriétés héritées de la classe UIObject

Le tableau suivant énumère les propriétés de la classe ComboBox héritées de la classe UIObject. Lors de l'accès à ces propriétés à partir de l'objet ComboBox, utilisez le formulaire *`comboBoxInstance.propertyName`*.

Propriété	Description
`UIObject.bottom`	Lecture seule ; position du bord inférieur de l'objet par rapport au bord inférieur de son parent.
`UIObject.height`	Lecture seule ; hauteur de l'objet, en pixels.
`UIObject.left`	Lecture seule ; bord gauche de l'objet, en pixels.
`UIObject.right`	Lecture seule. Position du bord droit de l'objet par rapport au bord droit de son parent.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Lecture seule ; renvoie la position du bord supérieur de l'objet par rapport à son parent.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Lecture seule ; largeur de l'objet, en pixels.
`UIObject.x`	Lecture seule ; bord gauche de l'objet, en pixels.
`UIObject.y`	Lecture seule ; bord supérieur de l'objet, en pixels.

Propriétés héritées de la classe UIComponent

Le tableau suivant énumère les propriétés de la classe ComboBox héritées de la classe UIComponent. Lors de l'accès à ces propriétés à partir de l'objet ComboBox, utilisez le formulaire *comboBoxInstance.propertyName*.

Propriété	Description
UIComponent.enabled	Indique si le composant peut recevoir le focus et la saisie.
UIComponent.tabIndex	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Evénements de la classe ComboBox

Le tableau suivant présente les événements de la classe ComboBox.

Evénement	Description
ComboBox.change	Diffusé lorsque la valeur de la liste déroulante change suite à l'interaction d'un utilisateur.
ComboBox.close	Diffusé lorsque la liste du composant ComboBox commence à se rétracter.
ComboBox.enter	Diffusé lorsque la touche Entrée est enfoncée.
ComboBox.itemRollOut	Diffusé lorsque le pointeur quitte un élément d'une liste déroulante.
ComboBox.itemRollOver	Diffusé lorsqu'un élément de liste déroulante est survolé.
ComboBox.open	Diffusé lorsque la liste déroulante commence à s'ouvrir.
ComboBox.scroll	Diffusé lorsque la liste déroulante est parcourue par l'utilisateur.

Evénements hérités de la classe UIObject

Le tableau suivant énumère les événements de la classe ComboBox hérités de la classe UIObject.

Evénement	Description
UIObject.draw	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
UIObject.hide	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
UIObject.load	Diffusé lorsque des sous-objets sont créés.
UIObject.move	Diffusé lorsque l'objet a été déplacé.

Evénement	Description
UIObject.resize	Diffusé lorsqu'un objet a été redimensionné.
UIObject.reveal	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
UIObject.unload	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant énumère les événements de la classe ComboBox hérités de la classe UIComponent.

Evénement	Description
UIComponent.focusIn	Diffusé lorsqu'un objet reçoit le focus.
UIComponent.focusOut	Diffusé lorsqu'un objet perd le focus.
UIComponent.keyDown	Diffusé lorsqu'une touche est enfoncée.
UIComponent.keyUp	Diffusé lorsqu'une touche est relâchée.

ComboBox.addItem()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
comboBoxInstance.addItem(label[, data])
comboBoxInstance.addItem({label:label[, data:data]})
comboBoxInstance.addItem(obj);
```

Paramètres

label Chaîne indiquant l'étiquette du nouvel élément.

data Données de l'élément (tout type). Ce paramètre est facultatif.

obj Objet possédant une propriété `label` et une propriété `data` facultative.

Valeur renvoyée

Index auquel l'élément a été ajouté.

Description

Méthode qui ajoute un nouvel élément à la fin de la liste.

Exemple

Utilisez une occurrence ComboBox appelée `my_cb` pour ajouter le code ActionScript suivant au panneau Actions pour la première image du scénario principal. Ce code ActionScript crée un composant ComboBox avec trois éléments ; chacun d'eux a une valeur data et une chaîne label. Lorsque vous testez le fichier SWF et que vous cliquez sur l'un des éléments, le panneau Sortie affiche l'identité de l'étiquette et de la valeur data cible :

```
// Ajout d'éléments au composant Combo Box.
my_cb.addItem("this is an Item");
my_cb.addItem({data:2, label:"second value"});
my_cb.addItem({data:3, label:"third value"});

// Ajout d'un écouteur d'événements et d'une fonction de
// gestionnaire d'événements.
var cbListener:Object = new Object();
cbListener.change = function(evt_obj:Object):Void  {
 var currentlySelected:Object = evt_obj.target.selectedItem;
 trace(evt_obj.target);
 trace("data: "+currentlySelected.data);
 trace("label: "+currentlySelected.label);
};
my_cb.addEventListener("change", cbListener);
```

ComboBox.addItemAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

comboBoxInstance.addItemAt(*index*, *label*[, *data*])

comboBoxInstance.addItemAt(*index*, {label:*label*[, data:*data*]})

comboBoxInstance.addItemAt(*index*, *obj*);

Paramètres

index Nombre 0 ou supérieur indiquant la position à laquelle insérer l'élément (index du nouvel élément).

label Chaîne indiquant l'étiquette du nouvel élément.

data Données de l'élément (tout type). Ce paramètre est facultatif.

obj Objet possédant des propriétés `label` et `data`.

Valeur renvoyée

Index auquel l'élément a été ajouté.

Description

Méthode : ajoute un nouvel élément à la fin de la liste, à l'emplacement d'index spécifié par le paramètre *index*. Les index supérieurs à `ComboBox.length` sont ignorés.

Exemple

Commencez avec une occurrence ComboBox appelée `my_cb` et une occurrence Button appelée `my_btn`. Ajoutez le code ActionScript suivant au panneau Actions pour la première image du scénario principal. Lorsque vous testez le fichier SWF, cliquez sur la liste déroulante pour afficher ces deux éléments. Cliquez ensuite sur le bouton et la prochaine fois que vous cliquerez sur la liste déroulante, vous verrez qu'un autre élément intitulé « first value » a été ajouté.

```
my_cb.addItem({data:2, label:"second value"});
my_cb.addItem({data:3, label:"third value"});

var btnListener:Object = new Object();
btnListener.click = function()  {
  my_cb.addItemAt(0, {data:1, label:"first value"});
};
my_btn.addEventListener("click", btnListener);
```

ComboBox.change

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.change = function(eventObject:Object) {
  // Votre code ici.
};
comboBoxInstance.addEventListener("change", listenerObject)
```

Description

Evénement ; diffusé à tous les écouteurs enregistrés lorsque la propriété `ComboBox.selectedIndex` ou `ComboBox.selectedItem` change suite à l'interaction d'un utilisateur.

Lors de l'utilisation d'un modèle d'événement dispatcher/écouteur, une occurrence du composant (*`comboBoxInstance`*) distribue un événement (dans ce cas, `change`) qui est géré par une fonction (également appelée *gestionnaire*) sur un objet écouteur (*`listenerObject`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée au déclenchement de l'événement. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`eventObject`*) à la méthode d'objet écouteur. Chaque objet événement est doté de propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `addEventListener()` (voir `EventDispatcher.addEventListener()`) sur l'occurrence du composant qui diffuse l'événement afin d'enregistrer l'écouteur avec cette occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

Lorsqu'une occurrence du composant ComboBox `my_cb` se trouve sur la scène, l'exemple suivant envoie le nom de l'occurrence de composant qui a généré l'événement `change` vers le panneau Sortie :

```
// Ajout de l'élément à la liste.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});

// Création d'un objet écouteur.
var cbListener:Object = new Object();

// Affectation d'une fonction à l'objet écouteur.
cbListener.change = function(event_obj:Object) {
 trace("Value changed to: "+event_obj.target.selectedItem.label);
};

// Ajout de l'écouteur.
my_cb.addEventListener("change", cbListener);
```

Voir aussi

`EventDispatcher.addEventListener()`

ComboBox.close()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

comboBoxInstance.close()

Paramètres

Aucun.

Valeur renvoyée

Aucune.

Description

Méthode qui ferme la liste déroulante.

Exemple

Lorsqu'une occurrence du composant ComboBox `my_cb` et qu'une occurrence du composant Button `my_button` se trouvent sur la scène, l'exemple suivant ferme la liste déroulante `my_cb` si vous cliquez sur le bouton `my_button` :

```
my_cb.addItem({data:2, label:"second value"});
my_cb.addItem({data:3, label:"third value"});

var btnListener:Object = new Object();
btnListener.click = function()  {
  my_cb.close();
};
my_button.addEventListener("click", btnListener);
```

Voir aussi

ComboBox.open()

ComboBox.close

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.close = function(eventObject:Object) {
  // Votre code ici.
};
comboBoxInstance.addEventListener("close", listenerObject)
```

Description

Evénement qui est diffusé à tous les écouteurs enregistrés lorsque la liste déroulante s'est entièrement rétractée.

Lors de l'utilisation d'un modèle d'événement dispatcher/écouteur, une occurrence du composant (*comboBoxInstance*) distribue un événement (dans ce cas, `close`) qui est géré par une fonction (également appelée *gestionnaire*) sur un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour associer l'écouteur à l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

Lorsqu'une occurrence du composant ComboBox `my_cb` se trouve sur la scène, l'exemple suivant envoie un message au panneau Sortie lorsque la liste déroulante est ouverte ou fermée :

```
// Ajout des éléments à la liste.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});
```

```
// Création d'un objet écouteur.
var cbListener:Object = new Object();
cbListener.open = function(evt_obj:Object) {
 trace("The ComboBox has opened.");
}
cbListener.close = function(evt_obj:Object){
   trace("The ComboBox has closed.");
}

// Ajout de l'écouteur.
my_cb.addEventListener("open", cbListener);
my_cb.addEventListener("close", cbListener);

// Ouverture de la liste déroulante.
my_cb.open();
```

Voir aussi

`EventDispatcher.addEventListener()`

ComboBox.dataProvider

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`comboBoxInstance``.dataProvider`

Description

Propriété : modèle de données pour les éléments affichés dans une liste. La valeur de cette propriété peut être un tableau ou tout objet qui implémente l'API DataProvider. La valeur par défaut est `[]`. Les composants List et ComboBox partagent la propriété `dataProvider` et toute modification de cette propriété est immédiatement disponible pour les deux composants.

Le composant List, comme d'autres composants de données, ajoute des méthodes au prototype de l'objet tableau pour être conforme à l'API DataProvider (voir DataProvider.as pour plus d'informations). Tout tableau qui existe parallèlement en tant que liste a donc automatiquement toutes les méthodes (`addItem()`, `getItemAt()`, etc.) nécessaires pour être le modèle d'une liste et peut être utilisé pour diffuser les changements de modèle vers plusieurs composants.

Si le tableau contient des objets, on accède à la propriété `labelField` ou `labelFunction` pour déterminer les parties de l'élément à afficher. La valeur par défaut étant « `label` », si un champ de ce type existe, il est choisi pour être affiché. Dans le cas contraire, une liste de tous les champs séparés par une virgule est affichée.

REMARQUE	Si le tableau contient des chaînes et aucun objet à tous les emplacements d'index, la liste ne peut pas trier les éléments et conserver l'état de sélection. Tout tri entraîne la perte de la sélection.

Toutes les occurrences implémentant l'API DataProvider peuvent être choisies comme fournisseurs de données d'un composant List. Cela inclut les objets Flash Remoting RecordSet, les composants Firefly DataSet, etc.

Exemple

Cet exemple utilise un tableau de chaînes utilisé pour remplir la liste déroulante pour l'occurrence du composant ComboBox `my_cb` :

```
my_cb.dataProvider = [{data:1, label:"First Item"}, {data:2, label:"Second
  Item"}];
/* est identique à
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});
*/
```

Cet exemple crée un tableau fournisseur de données et l'affecte à la propriété `dataProvider` :

```
var myDP:Array = new Array();
list.dataProvider = myDP;

for (var i:Number = 0; i < accounts.length; i++) {
  // Ces modifications apportées à DataProvider seront diffusées dans
  // la liste.
  myDP.addItem({label: accounts[i].name,
                data:  accounts[i].accountID});
}
```

ComboBox.dropdown

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

comboBoxInstance.dropdown

Description

Propriété (lecture seule) qui renvoie une référence à la liste du composant ComboBox. Le sous-composant List n'est instancié dans le composant ComboBox que lorsqu'il doit être affiché. En revanche, dès que vous accédez à la propriété `dropdown`, la liste est créée.

Exemple

Lorsqu'une occurrence du composant ComboBox my_cb se trouve sur la scène et que deux symboles de clip de la bibliothèque ont des valeurs d'identifiant de liaison définies sur `dw_id` et `fl_id`, le code ActionScript suivant utilise la propriété dropdown pour ajouter des icônes à chaque élément de la liste déroulante :

```
// Définition de la largeur de la propriété dropdown en fonction des tailles
// de l'étiquette.
my_cb.dropdownWidth = 200;

// Définition du style iconField dans la propriété dropdown du
// composant ComboBox.
// La propriété dropdown est une référence au composant List dans
// le composant ComboBox
// de façon à pouvoir définir les styles List pour le CB.
my_cb.dropdown.setStyle("iconField", "pIcon");

// Ajout des éléments à la liste.
my_cb.addItem({label:"Dreamweaver 1", pIcon:"dw_id"});
my_cb.addItem({label:"Flash 1", pIcon:"fl_id"});
my_cb.addItem({label:"Flash 2", pIcon:"fl_id"});
```

Voir aussi

`ComboBox.dropdownWidth`

ComboBox.dropdownWidth

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

comboBoxInstance.dropdownWidth

Description

Propriété : limite de la largeur de la liste déroulante, en pixels. La valeur par défaut est la largeur du composant ComboBox (l'occurrence TextInput plus l'occurrence SimpleButton).

Exemple

Lorsqu'une occurrence du composant ComboBox `my_cb` se trouve sur la scène, le code ActionScript suivant définit la largeur de la liste déroulante en fonction des étiquettes :

```
// Définition de la largeur de la propriété dropdown en fonction des
// tailles de l'étiquette.
my_cb.dropdownWidth = 200;

// Ajout des éléments à la liste.
my_cb.addItem("ComboBox");
my_cb.addItem({data:2, label:"This is a long label"});
my_cb.addItem({data:3, label:"This has an even longer label"});
```

Voir aussi

ComboBox.dropdown

ComboBox.editable

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

comboBoxInstance.editable

Description

Propriété : indique si la liste déroulante est modifiable (`true`) ou non (`false`). Un composant ComboBox modifiable peut comporter des valeurs saisies par l'utilisateur dans le champ de texte mais qui n'apparaissent pas dans la liste déroulante. Lorsqu'une liste déroulante n'est pas modifiable, toute saisie est impossible dans le champ de texte. Ce champ affiche le texte de l'élément de la liste. La valeur par défaut est `false`.

Le fait de rendre une liste déroulante modifiable efface le contenu de son champ de texte. L'opération définit également l'index (et l'élément) sélectionné sur `undefined`. Pour rendre une liste déroulante modifiable tout en conservant l'élément sélectionné, utilisez le code suivant :

```
var ix:Number = myComboBox.selectedIndex;
myComboBox.editable = true; // Suppression du champ de texte.
myComboBox.selectedIndex = ix; // Recopie de l'étiquette dans le champ
                               // de texte.
```

Exemple

Lorsqu'une occurrence du composant ComboBox `my_cb` se trouve sur la scène, le code ActionScript suivant crée une liste déroulante et deux écouteurs. Le premier écouteur gère la sélection de l'étiquette « Add new item » pour rendre un champ de la liste déroulante modifiable. Le second écouteur gère la sélection de la touche Entrée par l'utilisateur pour ajouter son entrée à la liste déroulante :

```
// Ajout d'éléments à la liste déroulante.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});
my_cb.addItem({data:-1, label:"Add new item..."});

// Réponse à l'utilisateur ayant cliqué sur « Add new item ».
function changeListener(evt_obj:Object) {
 if (evt_obj.target.selectedItem.data == -1) {
  evt_obj.target.editable = true;
 } else if (evt_obj.target.selectedIndex != undefined) {
  evt_obj.target.editable = false;
  evt_obj.target.setFocus();
 }
}
my_cb.addEventListener("change", changeListener);

// Réponse à l'utilisateur ayant appuyé sur la touche Entrée après
// avoir ajouté un nouveau nom d'élément.
function enterListener(evt_obj:Object) {
 if (evt_obj.target.value != '') {
  evt_obj.target.addItem({data:'', label:evt_obj.target.value});
 }
```

```
 evt_obj.target.editable = false;
 evt_obj.target.selectedIndex = evt_obj.target.dataProvider.length-1;
 evt_obj.target.setFocus();
}
my_cb.addEventListener("enter", enterListener);
```

ComboBox.enter

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.enter = function(eventObject:Object) {
  // Votre code ici.
};
comboBoxInstance.addEventListener("enter", listenerObject)
```

Description

Evénement qui est diffusé à tous les écouteurs enregistrés lorsque l'utilisateur appuie sur la touche Entrée dans le champ de texte. Il s'agit d'un événement TextInput qui est diffusé uniquement à partir de listes déroulantes modifiables. Pour plus d'informations, voir `TextInput.enter`.

Lors de l'utilisation d'un modèle d'événement dispatcher/écouteur, une occurrence du composant (*`comboBoxInstance`*) distribue un événement (dans ce cas, `enter`) qui est géré par une fonction (également appelée *gestionnaire*) sur un objet écouteur (*`listenerObject`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`eventObject`*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour associer l'écouteur à l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

Lorsqu'une occurrence du composant ComboBox `my_cb` se trouve sur la scène, le code ActionScript suivant crée une liste déroulante et deux écouteurs. Le premier écouteur gère la sélection de l'étiquette « Add new item » (Ajouter un nouvel élément) pour rendre un champ de la liste déroulante modifiable. Le second écouteur gère la pression de la touche Entrée par l'utilisateur pour ajouter son entrée à la liste déroulante :

```
// Ajout d'éléments à la liste déroulante.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});
my_cb.addItem({data:-1, label:"Add new item..."});

// Réponse à l'utilisateur ayant cliqué sur « Add new item ».
function changeListener(evt_obj:Object) {
 if (evt_obj.target.selectedItem.data == -1) {
  evt_obj.target.editable = true;
 } else if (evt_obj.target.selectedIndex != undefined) {
  evt_obj.target.editable = false;
  evt_obj.target.setFocus();
 }
}
my_cb.addEventListener("change", changeListener);

// Réponse à l'utilisateur ayant appuyé sur la touche Entrée après
// avoir ajouté un nouveau nom d'élément.
function enterListener(evt_obj:Object) {
 if (evt_obj.target.value != '') {
  evt_obj.target.addItem({data:'', label:evt_obj.target.value});
 }
 evt_obj.target.editable = false;
 evt_obj.target.selectedIndex = evt_obj.target.dataProvider.length-1;
 evt_obj.target.setFocus();
}
my_cb.addEventListener("enter", enterListener);
```

Voir aussi

`EventDispatcher.addEventListener()`

ComboBox.getItemAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
comboBoxInstance.getItemAt(index)
```

Paramètres

index Index de l'élément à récupérer. Nombre supérieur ou égal à 0 et inférieur à la valeur de `ComboBox.length`.

Valeur renvoyée

La valeur ou l'objet d'élément indexé. La valeur n'est pas définie si l'index est hors limites.

Description

Méthode qui récupère l'élément à l'emplacement d'index spécifié.

Exemple

Lorsqu'une occurrence du composant ComboBox `my_cb` se trouve sur la scène, le code ActionScript suivant affiche l'étiquette du premier élément de la liste déroulante dans le panneau Sortie :

```
// Ajout de l'élément à la liste.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});

trace(my_cb.getItemAt(1).label);
```

ComboBox.itemRollOut

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.itemRollOut = function(eventObject:Object) {
  // Votre code ici.
};
comboBoxInstance.addEventListener("itemRollOut", listenerObject)
```

Objet événement

Outre les propriétés standard de l'objet événement, l'événement `itemRollOut` possède une propriété `index`. L'index correspond au numéro de l'élément survolé par le pointeur.

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque le pointeur cesse de survoler un élément de la liste déroulante. Il s'agit d'un événement List diffusé à partir d'une liste déroulante. Pour plus d'informations, voir `List.itemRollOut`.

Lors de l'utilisation d'un modèle d'événement dispatcher/écouteur, une occurrence du composant (*`comboBoxInstance`*) distribue un événement (dans ce cas, `itemRollOut`) qui est géré par une fonction (également appelée *gestionnaire*) sur un objet écouteur (*`listenerObject`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`eventObject`*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Pour finir, vous appelez la méthode `addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour associer l'écouteur à l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Exemple

Lorsqu'une occurrence du composant ComboBox `my_cb` se trouve sur la scène, le code ActionScript suivant envoie un message au panneau Sortie indiquant l'index de l'élément et l'événement lorsque le pointeur survole un élément ou cesse de le survoler :

```
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});

// Création d'un objet écouteur.
var cbListener:Object = new Object();
cbListener.itemRollOver = function(evt_obj:Object) {
 trace("index: "+evt_obj.index+", event: "+evt_obj.type);
};
cbListener.itemRollOut = function(evt_obj:Object) {
 trace("index: "+evt_obj.index+", event: "+evt_obj.type);
};

// Ajout de l'écouteur.
my_cb.addEventListener("itemRollOver", cbListener);
my_cb.addEventListener("itemRollOut", cbListener);
```

Voir aussi

`ComboBox.itemRollOver`, `EventDispatcher.addEventListener()`

ComboBox.itemRollOver

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.itemRollOver = function(eventObject:Object) {
  // Votre code ici.
};
comboBoxInstance.addEventListener("itemRollOver", listenerObject)
```

Objet événement

Outre les propriétés standard de l'objet événement, l'événement `itemRollOver` possède une propriété `index`. L'index correspond au numéro de l'élément survolé par le pointeur.

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque le pointeur survole un élément de la liste déroulante. Il s'agit d'un événement List diffusé à partir d'une liste déroulante. Pour plus d'informations, voir `List.itemRollOver`.

Lors de l'utilisation du modèle d'événement dispatcher/écouteur, une occurrence du composant (*`comboBoxInstance`*) distribue un événement (dans ce cas, `itemRollOver`) qui est géré par une fonction (également appelée *gestionnaire*) sur un objet écouteur (*`listenerObject`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`eventObject`*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Pour finir, vous appelez la méthode `addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour associer l'écouteur à l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Exemple

Lorsqu'une occurrence du composant ComboBox `my_cb` se trouve sur la scène, le code ActionScript suivant envoie un message au panneau Sortie indiquant l'index de l'élément et l'événement lorsque le pointeur survole un élément ou cesse de le survoler :

```
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});

// Création d'un objet écouteur.
var cbListener:Object = new Object();
cbListener.itemRollOver = function(evt_obj:Object) {
 trace("index: " + evt_obj.index + ", event: " + evt_obj.type);
};
cbListener.itemRollOut = function(evt_obj:Object) {
 trace("index: " + evt_obj.index + ", event: " + evt_obj.type);
};

// Ajout de l'écouteur.
my_cb.addEventListener("itemRollOver", cbListener);
my_cb.addEventListener("itemRollOut", cbListener);
```

Voir aussi

`ComboBox.itemRollOut`, `EventDispatcher.addEventListener()`

ComboBox.labelField

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`comboBoxInstance``.labelField`

Description

Propriété : nom du champ dans les objets du tableau `dataProvider` à utiliser comme champ d'étiquette. Il s'agit de la propriété du composant List qui est disponible à partir d'une occurrence de composant ComboBox. Pour plus d'informations, voir `List.labelField`.

La valeur par défaut est `undefined`.

Exemple

L'exemple suivant définit la propriété `dataProvider` sur un tableau de chaînes ainsi que la propriété `labelField` afin d'indiquer que le champ `name` doit être utilisé comme étiquette pour la liste déroulante :

```
my_cb.dataProvider = [
  {name:"Gary", gender:"male"},
  {name:"Susan", gender:"female"} ];

my_cb.labelField = "name";
```

Voir aussi

`List.labelFunction`

ComboBox.labelFunction

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`comboBoxInstance``.labelFunction`

Description

Propriété : fonction qui calcule l'étiquette d'un élément de fournisseur de données. Vous devez définir cette fonction. La valeur par défaut est `undefined`.

Exemple

L'exemple suivant crée un fournisseur de données, puis définit une fonction afin de spécifier l'objet à utiliser comme étiquette dans la liste déroulante :

```
myComboBox.dataProvider  = [
  {firstName:"Nigel", lastName:"Pegg", age:"really young"},
  {firstName:"Gary", lastName:"Grossman", age:"young"},
  {firstName:"Chris", lastName:"Walcott", age:"old"},
  {firstName:"Greg", lastName:"Yachuk", age:"really old"} ];

myComboBox.labelFunction = function(itemObj){
  return (itemObj.lastName + ", " + itemObj.firstName);
}
```

Voir aussi

`List.labelField`

ComboBox.length

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
comboBoxInstance.length
```

Description

Propriété (lecture seule) : longueur de la liste déroulante. Il s'agit de la propriété du composant List qui est disponible à partir d'une occurrence ComboBox. Pour plus d'informations, voir `List.length`. La valeur par défaut est 0.

Exemple

L'exemple suivant stocke la valeur de `length` dans une variable :

```
var dropdownItemCount:Number = myComboBox.length;
```

ComboBox.open()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
comboBoxInstance.open()
```

Paramètres

Aucun.

Valeur renvoyée

Aucune.

Description

Méthode qui ouvre la liste déroulante.

Exemple

Lorsqu'une occurrence du composant ComboBox `my_cb` et qu'une occurrence du composant Button `my_button` se trouvent sur la scène, l'exemple suivant ouvre la liste déroulante `my_cb` si vous cliquez sur le bouton `my_button` :

```
my_cb.addItem({data:2, label:"second value"});
my_cb.addItem({data:3, label:"third value"});

var btnListener:Object = new Object();
btnListener.click = function()  {
  my_cb.open();
};
my_button.addEventListener("click", btnListener);
```

Voir aussi

`ComboBox.close()`

ComboBox.open

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.open = function(eventObject:Object) {
  // Votre code ici.
};
comboBoxInstance.addEventListener("open", listenerObject)
```

Description

Evénement qui est diffusé à tous les écouteurs enregistrés lorsque la liste déroulante est entièrement ouverte.

Lors de l'utilisation du modèle d'événement dispatcher/écouteur, une occurrence du composant (*comboBoxInstance*) distribue un événement (dans ce cas, `open`) qui est géré par une fonction (également appelée *gestionnaire*) sur un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Pour finir, vous appelez la méthode `addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour associer l'écouteur à l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Exemple

Lorsqu'une occurrence du composant ComboBox `my_cb` se trouve sur la scène, l'exemple suivant envoie un message au panneau Sortie lorsque la liste déroulante est ouverte ou fermée :

```
// Ajout des éléments à la liste.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});

// Création d'un objet écouteur.
var cbListener:Object = new Object();
cbListener.open = function(evt_obj:Object) {
 trace("The ComboBox has opened.");
}
cbListener.close = function(evt_obj:Object){
   trace("The ComboBox has closed.");
}

// Ajout de l'écouteur.
my_cb.addEventListener("open", cbListener);
my_cb.addEventListener("close", cbListener);

// Ouverture de la liste déroulante.
my_cb.open();
```

Voir aussi

`ComboBox.close`, `EventDispatcher.addEventListener()`

ComboBox.removeAll()

Disponibilité
Flash Player 6 (6.0.79.0).

Edition
Flash MX 2004.

Utilisation
comboBoxInstance.removeAll()

Paramètres
Aucun.

Valeur renvoyée
Aucune.

Description
Méthode qui supprime tous les éléments de la liste. Il s'agit d'une méthode du composant List qui est disponible à partir d'une occurrence de composant ComboBox.

Exemple
Lorsqu'une occurrence du composant ComboBox `my_cb` et qu'une occurrence du composant Button `clear_button` se trouvent sur la scène, le code ActionScript suivant positionne la liste déroulante et le bouton l'un à côté de l'autre. Lorsque vous cliquez sur la liste déroulante, une liste d'éléments apparaît. Lorsque vous cliquez sur le bouton, les éléments de la liste déroulante sont effacés :

```
my_cb.move(10, 10);
clear_button.move(120, 10);

// Création d'un dataprovider.
var myDP_array:Array = new Array();
myDP_array.push({data:1, label:"First Item"});
myDP_array.push({data:2, label:"Second Item"});

my_cb.dataProvider = myDP_array;

// Définition d'un objet écouteur d'événements.
var clearListener:Object = new Object();
clearListener.click = function(evt_obj:Object){
 my_cb.removeAll();
}

// Ajout de l'écouteur.
clear_button.addEventListener("click", clearListener);
```

Voir aussi
ComboBox.removeItemAt(), ComboBox.replaceItemAt()

ComboBox.removeItemAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

comboBoxInstance.removeItemAt(*index*)

Paramètres

index Nombre indiquant la position de l'élément à supprimer. L'index commence à zéro.

Valeur renvoyée

Un objet : l'élément supprimé (« undefined » si aucun élément n'existe).

Description

Méthode qui supprime un élément à l'emplacement d'index indiqué. Un index disparaît parmi les index de la liste situés après l'index indiqué par le paramètre *index*. Il s'agit d'une méthode du composant List qui est disponible à partir d'une occurrence de composant ComboBox.

Exemple

Lorsqu'une occurrence du composant ComboBox `my_cb` et qu'une occurrence du composant Button `clear_button` se trouvent sur la scène, le code ActionScript suivant positionne la liste déroulante et le bouton l'un à côté de l'autre. Lorsque vous cliquez sur la liste déroulante, une liste de deux éléments apparaît. Lorsque vous cliquez sur le bouton, le second élément de la liste déroulante est effacé (à la position d'index `1`, car la valeur est basée sur zéro) :

```
my_cb.move(10, 10);
clear_button.move(120, 10);

// Création d'un dataprovider.
var myDP_array:Array = new Array();
myDP_array.push({data:1, label:"First Item"});
myDP_array.push({data:2, label:"Second Item"});

my_cb.dataProvider = myDP_array;
```

```
// Définition d'un objet écouteur d'événements.
var clearListener:Object = new Object();
clearListener.click = function(evt_obj:Object){
 my_cb.removeItemAt(1);
}

// Ajout de l'écouteur.
clear_button.addEventListener("click", clearListener);
```

Voir aussi

`ComboBox.removeAll()`, `ComboBox.replaceItemAt()`

ComboBox.replaceItemAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

comboBoxInstance.replaceItemAt(*index*, *label*[, *data*])

comboBoxInstance.replaceItemAt(*index*, {label:*label*[, data:*data*]})

comboBoxInstance.replaceItemAt(*index*, *obj*);

Paramètres

index Nombre 0 ou supérieur indiquant la position à laquelle insérer l'élément (index du nouvel élément).

label Chaîne indiquant l'étiquette du nouvel élément.

data Données de l'élément. Ce paramètre est facultatif.

obj Objet possédant des propriétés `label` et `data`.

Valeur renvoyée

Aucune.

Description

Méthode qui remplace le contenu de l'élément à l'index spécifié. Il s'agit d'une méthode du composant List qui est disponible à partir d'une occurrence de composant ComboBox.

Exemple

Lorsqu'une occurrence du composant ComboBox `my_cb` et une occurrence du composant TextInput `label_ti` se trouvent sur la scène, le code ActionScript suivant ajoute la saisie de l'utilisateur à la liste déroulante lorsque l'utilisateur appuie sur la touche Entrée :

```
// Ajout des éléments à la liste.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});

// Création de l'écouteur pour l'utilisateur appuyant sur la
// touche Entrée dans le champ de saisie de texte.
var tiListener:Object = new Object();
tiListener.enter = function(evt_obj:Object) {
 my_cb.replaceItemAt(my_cb.selectedIndex, {label:evt_obj.target.text});
 // Nécessaire pour actualiser l'entrée ComboBox modifiée récemment.
 my_cb.selectedIndex = my_cb.selectedIndex;
};
label_ti.addEventListener("enter", tiListener);
```

Voir aussi

`ComboBox.removeAll()`, `ComboBox.removeItemAt()`

ComboBox.restrict

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`comboBoxInstance``.restrict`

Description

Propriété qui désigne le jeu de caractères qu'un utilisateur peut saisir dans le champ de texte d'une liste déroulante. La valeur par défaut est `undefined`. Si la valeur de cette propriété est `null` ou une chaîne vide (`""`), un utilisateur peut entrer n'importe quel caractère. Si elle contient une chaîne de caractères, l'utilisateur ne peut saisir que les caractères de cette chaîne. Cette dernière est lue de gauche à droite. Vous pouvez spécifier une plage en utilisant un tiret (`-`).

Si la chaîne commence par un caret (`^`), tous les caractères qui le suivent sont refusés. Si la chaîne ne commence pas par un caret, ses caractères sont acceptés.

Vous pouvez utiliser la barre oblique inverse (\) pour entrer un trait d'union (-), un caret (^) ou une barre oblique inverse (\), comme suit :

```
\^
\-
\\
```

La saisie d'une barre oblique inverse entre guillemets dans le panneau Actions a une signification particulière pour l'interpréteur de guillemets du panneau. Cette syntaxe indique en effet que le caractère suivant la barre oblique inversée doit être traité « tel quel ». Le code suivant, par exemple, permet d'entrer une apostrophe :

```
var leftQuote = "\'";
```

L'interpréteur .restrict du panneau Actions utilise également la barre oblique inverse comme caractère d'échappement. Vous pouvez donc supposer que la chaîne suivante est correcte :

```
myText.restrict = "0-9\-\^\\";
```

Toutefois, cette expression étant placée entre guillemets, la valeur `0-9-^\` est envoyée à l'interpréteur .restrict, et ce dernier ne la comprend pas.

L'interpréteur .restrict exigeant l'utilisation de guillemets, vous devez non seulement lui fournir l'expression appropriée, mais également utiliser le caractère d'échappement pour que cette expression soit traitée correctement par l'interpréteur de guillemets intégré du panneau Actions. Pour envoyer la valeur `0-9\-\^\\` à l'interpréteur restrict, vous devez donc entrer le code suivant :

```
myCombo.restrict = "0-9\\-\\^\\\\";
```

La propriété `restrict` limite uniquement l'interaction de l'utilisateur, un script pouvant insérer n'importe quel texte dans le champ de texte. Cette propriété ne se synchronise pas avec les cases à cocher de polices vectorielles intégrées de l'inspecteur des propriétés.

Exemple

Avec une occurrence du composant ComboBox `my_cb`, le code ActionScript suivant limite la saisie de caractères aux chiffres 0 à 9, aux tirets et aux points :

```
// Ajout des éléments à la liste.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});

// Activation de la modification de la liste déroulante.
my_cb.editable = true;

// Limitation des caractères pouvant être saisis dans ComboBox.
my_cb.restrict = "0-9\\-\\.\\ ";
```

Dans l'exemple suivant, la première ligne de code limite le champ de texte aux lettres majuscules, aux nombres et aux espaces. La seconde ligne autorise tous les caractères, à l'exception des lettres minuscules.

```
my_combo.restrict = "A-Z 0-9";
my_combo.restrict = "^a-z";
```

Le code suivant permet à un utilisateur d'entrer les caractères « 0 1 2 3 4 5 6 7 8 9 - ^ \ » dans l'occurrence `myCombo`. Vous devez utiliser une double barre oblique inverse pour activer la fonction d'échappement pour les caractères `-`, `^` et `\`. La première barre oblique inverse `\` échappe les guillemets doubles et la seconde `\` indique à l'interpréteur que le caractère suivant ne doit pas être traité comme un caractère spécial.

```
myCombo.restrict = "0-9\\-\\^\\\\";
```

ComboBox.rowCount

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`comboBoxInstance``.rowCount`

Description

Propriété ; nombre maximum de lignes visibles dans la liste déroulante avant que cette dernière n'insère une barre de défilement. La valeur par défaut est 5.

Si le nombre d'éléments de la liste déroulante est supérieur à la propriété `rowCount`, la liste est redimensionnée et une barre de défilement est affichée si nécessaire. Si la liste déroulante contient moins d'éléments que la valeur indiquée dans la propriété `rowCount`, elle est redimensionnée en fonction du nombre d'éléments contenus.

Ce comportement diffère de celui du composant List, qui affiche toujours le nombre de lignes spécifié par sa propriété `rowCount`, même en cas d'espaces vides.

Si la valeur est négative ou décimale, le comportement n'est pas défini.

Exemple

Avec une occurrence du composant ComboBox `my_cb`, le code ActionScript suivant définit la liste déroulante pour qu'elle affiche les trois premiers éléments et ajoute une barre de défilement pour afficher le quatrième :

```
// Ajout des éléments à la liste.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});
my_cb.addItem({data:3, label:"Third Item"});
my_cb.addItem({data:4, label:"Fourth Item"});

// Affichage de la barre de défilement si ComboBox comporte plus
// de 3 éléments.
my_cb.rowCount = 3;
```

ComboBox.scroll

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.scroll = function(eventObject:Object) {
  // Votre code ici.
};
comboBoxInstance.addEventListener("scroll", listenerObject);
```

Objet événement

Outre les propriétés standard de l'objet événement, l'événement scroll possède une propriété supplémentaire : `direction`. Il s'agit d'une chaîne ayant deux valeurs possibles : `"horizontal"` ou `"vertical"`. Pour un événement `scroll` ComboBox, la valeur est toujours « `vertical` ».

Description

Evénement qui est diffusé à tous les écouteurs enregistrés lorsque la liste déroulante défile. Il s'agit d'un événement de composant List qui est disponible pour le composant ComboBox.

Lors de l'utilisation d'un modèle d'événement dispatcher/écouteur, une occurrence du composant (*comboBoxInstance*) distribue un événement (dans ce cas, `scroll`) qui est géré par une fonction (également appelée *gestionnaire*) sur un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Pour finir, vous appelez la méthode `addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour associer l'écouteur à l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Exemple

Avec une occurrence du composant ComboBox `my_cb`, l'exemple suivant envoie un message au panneau Sortie indiquant l'index de l'élément que la liste fait défiler :

```
// Ajout des éléments à la liste.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});
my_cb.addItem({data:3, label:"Third Item"});
my_cb.addItem({data:4, label:"Fourth Item"});

// Affichage de la barre de défilement si ComboBox comporte plus
// de 2 éléments.
my_cb.rowCount = 3;

// Création d'un objet écouteur.
var cbListener:Object = new Object();
cbListener.scroll = function(evt_obj:Object) {
 trace("The list had been scrolled to item # "+evt_obj.position);
};

// Ajout de l'écouteur.
my_cb.addEventListener("scroll", cbListener);
```

Voir aussi

`EventDispatcher.addEventListener()`

ComboBox.selectedIndex

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

comboBoxInstance.selectedIndex

Description

Propriété : numéro d'index de l'élément sélectionné dans la liste déroulante. La valeur par défaut est 0. L'affectation de cette propriété annule la sélection en cours, sélectionne l'élément indiqué et affiche l'étiquette de l'élément dans le champ de texte du composant ComboBox.

Si vous affectez une valeur qui sort de la plage à cette propriété, Flash l'ignore. Toute saisie dans le champ de texte d'une liste déroulante modifiable définit `selectedIndex` sur `undefined`.

Exemple

Avec une occurrence du composant ComboBox `my_cb`, le code suivant sélectionne le dernier élément de la liste (autrement, par défaut, il afficherait le premier élément) :

```
// Ajout des éléments à la liste.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});
my_cb.addItem({data:3, label:"Third Item"});
my_cb.addItem({data:4, label:"Fourth Item"});

// Sélection du dernier élément de la liste.
my_cb.selectedIndex = my_cb.length-1;
```

Voir aussi

ComboBox.selectedItem

ComboBox.selectedItem

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

comboBoxInstance.selectedItem

Description

Propriété : valeur de l'élément sélectionné dans la liste déroulante.

Si la liste déroulante est modifiable, `selectedItem` renvoie la valeur `undefined` lorsque l'utilisateur entre un texte dans la zone de texte. La propriété n'a une valeur que si vous sélectionnez un élément dans la liste déroulante ou définissez la valeur via ActionScript. Si la liste déroulante est statique, la valeur `selectedItem` est toujours valide et renvoie `undefined` lorsque la liste ne contient pas d'élément.

Exemple

Avec une occurrence du composant ComboBox `my_cb`, l'exemple suivant affiche les valeurs des propriétés data et label `selectedItem` :

```
// Ajout des éléments à la liste.
my_cb.addItem({data:1, label:"First Item"});
my_cb.addItem({data:2, label:"Second Item"});
my_cb.addItem({data:3, label:"Third Item"});
my_cb.addItem({data:4, label:"Fourth Item"});

var cbListener:Object = new Object();
cbListener.change = function(evt_obj:Object) {
 var item_obj:Object = my_cb.selectedItem;
 var i:String;
 for (i in item_obj) {
  trace(i + ":\t" + item_obj[i]);
 }
 trace("");
};
my_cb.addEventListener("change", cbListener);
```

Voir aussi

`ComboBox.dataProvider`, `ComboBox.selectedIndex`

ComboBox.sortItems()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
comboBoxInstance.sortItems([compareFunc], [optionsFlag])
```

Paramètres

compareFunc Référence à une fonction permettant de comparer deux éléments pour déterminer leur ordre de tri. Pour plus d'informations, reportez-vous à `Array.sort()` dans le *Guide de référence du langage ActionScript 2.0*. Ce paramètre est facultatif.

optionsFlag Permet d'effectuer plusieurs types de tris dans un même tableau sans avoir à copier ce dernier ou à le trier à plusieurs reprises. Ce paramètre est facultatif.

Les valeurs possibles pour *optionsFlag* sont les suivantes :

- `Array.DESCENDING` trie par ordre décroissant.
- `Array.CASEINSENSITIVE` trie sans respecter la casse.
- `Array.NUMERIC` trie par ordre numérique si les deux éléments comparés sont des nombres. S'il ne s'agit pas de nombres, effectuez une comparaison de type chaîne (qui peut être non sensible à la casse si l'indicateur est spécifié).
- `Array.UNIQUESORT` renvoie un code d'erreur (0) au lieu d'un tableau trié si deux objets sont identiques ou comportent des champs de tri identiques.
- `Array.RETURNINDEXEDARRAY` renvoie un tableau d'index de nombres entiers correspondant au résultat du tri. Par exemple, le tableau suivant renverra la seconde ligne de code et ne sera pas modifié :

```
["a", "d", "c", "b"]
[0, 3, 2, 1]
```

Vous pouvez combiner ces options en une seule valeur. Par exemple, le code suivant associe les options 3 et 1 :

```
array.sort (Array.NUMERIC | Array.DESCENDING)
```

Valeur renvoyée

Aucune.

Description

Méthode qui trie les éléments de la liste déroulante selon la fonction de comparaison ou les options de tri spécifiées.

Exemple

Cet exemple trie selon des étiquettes en majuscules. Les éléments `a` et `b` sont transmis à la fonction et possèdent les champs `label` et `data` :

```
myComboBox.sortItems(upperCaseFunc);
function upperCaseFunc(a,b){
    return a.label.toUpperCase() > b.label.toUpperCase();
}
```

L'exemple suivant utilise la fonction `upperCaseFunc()` définie ci-dessus avec le paramètre *`optionsFlag`* pour trier les éléments d'une occurrence ComboBox nommée `myComboBox` :

```
myComboBox.addItem("Mercury");
myComboBox.addItem("Venus");
myComboBox.addItem("Earth");
myComboBox.addItem("planet");
myComboBox.sortItems(upperCaseFunc, Array.DESCENDING);
// L'ordre de tri obtenu de myComboBox sera :
// Venus
// planet
// Mercury
// Earth
```

ComboBox.sortItemsBy()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`comboBoxInstance``.sortItemsBy(`*`fieldName`*`, `*`order`*` [`*`optionsFlag`*`])`

Paramètres

`fieldName` Chaîne spécifiant le nom du champ à utiliser pour le tri. Cette valeur est en général « `label` » ou « `data` ».

`ordre` Chaîne qui indique si le tri des éléments doit être effectué dans l'ordre croissant (`"ASC"`) ou décroissant (`"DESC"`).

optionsFlag Permet d'effectuer plusieurs types de tris dans un même tableau sans avoir à copier ce dernier ou à le trier à plusieurs reprises. Ce paramètre est facultatif, mais s'il est utilisé, il doit remplacer le paramètre *ordre*.

Les valeurs possibles pour *optionsFlag* sont les suivantes :

- `RAray.DESCENDING` trie par ordre décroissant.
- `RAray.CASEINSENSITIVE` trie sans respecter la casse.
- `RAray.NUMERIC` trie par ordre numérique si les deux éléments comparés sont des nombres. S'il ne s'agit pas de nombres, effectuez une comparaison de type chaîne (qui peut être non sensible à la casse si l'indicateur est spécifié).
- `RAray.UNIQUESORT` renvoie un code d'erreur (0) au lieu d'un tableau trié si deux objets sont identiques ou comportent des champs de tri identiques.
- `RAray.RETURNINDEXEDARRAY` renvoie un tableau d'index de nombres entiers correspondant au résultat du tri. Par exemple, le tableau suivant renverra la seconde ligne de code et ne sera pas modifié :

```
["a", "d", "c", "b"]
[0, 3, 2, 1]
```

Vous pouvez combiner ces options en une seule valeur. Par exemple, le code suivant associe les options 3 et 1 :

```
array.sort (Array.NUMERIC | Array.DESCENDING)
```

Valeur renvoyée

Aucune.

Description

Méthode qui trie les éléments de la liste déroulante par ordre alphabétique ou numérique, dans l'ordre spécifié, avec le nom de champ spécifié. Si les éléments *fieldName* sont une combinaison de chaînes de texte et de nombres entiers, ce sont les éléments entiers qui sont indiqués en premier. Le paramètre *fieldName* est généralement « `label` » ou « `data` », mais les programmeurs expérimentés peuvent spécifier n'importe quelle valeur primitive.
Vous pouvez également utiliser le paramètre *optionsFlag* pour définir un type de tri.

Exemple

Les exemples suivants sont basés sur une occurrence ComboBox nommée `myComboBox`, qui contient quatre éléments appelés « `Apples` », « `Bananas` », « `cherries` » et « `Grapes` » :

```
// Commencez par alimenter le composant ComboBox avec les éléments.
myComboBox.addItem("Bananas");
myComboBox.addItem("Apples");
myComboBox.addItem("cherries");
myComboBox.addItem("Grapes");

// L'instruction suivante effectue un tri avec le paramètre ordre défini
// sur « ASC »,
// et dans le résultat, « cherries » est placé au bas de la liste
// car le tri respecte la casse.
myComboBox.sortItemsBy("label", "ASC");
// Ordre du résultat : Apples, Bananas, Grapes, cherries

// L'instruction suivante effectue un tri avec le paramètre ordre défini
// sur « DESC »,
// et dans le résultat, « cherries » est placé en haut de la liste
// car le tri respecte la casse.
myComboBox.sortItemsBy("label", "DESC");
 // Ordre du résultat : cherries, Grapes, Bananas, Apples

// L'instruction suivante effectue un tri avec le paramètre
// optionsFlag défini sur Array.CASEINSENSITIVE.
// Remarquez que l'ordre croissant correspond au paramètre par défaut.
myComboBox.sortItemsBy("label", Array.CASEINSENSITIVE);
// Ordre du résultat : Apples, Bananas, cherries, Grapes

// L'instruction suivante effectue un tri avec le paramètre
// optionsFlag défini sur Array.CASEINSENSITIVE | Array.DESCENDING.
myComboBox.sortItemsBy("label", Array.CASEINSENSITIVE | Array.DESCENDING);
// Ordre du résultat : Grapes, cherries, Bananas, Apples
```

ComboBox.text

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

comboBoxInstance.text

Description

Propriété : texte du champ de texte. Vous pouvez obtenir et définir cette valeur pour les listes déroulantes modifiables. Pour les listes déroulantes statiques, la valeur est en lecture seule.

Exemple

L'exemple suivant définit la valeur `text` actuelle d'une liste déroulante modifiable :

```
my_cb.addItem("Arkansas");
my_cb.addItem("Georgia");

my_cb.editable = true;
my_cb.text = "California";
```

ComboBox.textField

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

comboBoxInstance.textField

Description

Propriété (lecture seule) : référence au composant TextInput contenu dans le composant ComboBox.

Cette propriété vous permet d'accéder au composant TextInput sous-jacent pour le manipuler. Par exemple, vous pouvez changer la sélection du champ de texte ou restreindre les caractères pouvant y être saisis.

Exemple

Le code suivant restreint la zone de texte de `myComboBox` à la saisie de chiffres comportant six caractères au maximum :

```
// Ajout des éléments à la liste.
my_cb.addItem({data:0xFFFFFF, label:"white"});
my_cb.addItem({data:0x000000, label:"black"});

my_cb.editable = true;

// Limitation des données pouvant être saisies dans le champ
// de texte aux chiffres compris entre 0 et 9.
my_cb.restrict = "0-9";

// Limitation de la saisie à 6 caractères.
my_cb.textField.maxChars = 6;
```

ComboBox.value

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`comboBoxInstance``.value`

Description

Propriété en lecture seule ; si la liste déroulante est modifiable, `value` renvoie l'étiquette de l'élément. Si la liste déroulante est statique, `value` renvoie les données de l'élément.

Exemple

L'exemple suivant insère les données dans la liste déroulante en définissant la propriété `dataProvider`. Il affiche ensuite `value` dans le panneau Sortie. Pour finir, il sélectionne « `California` » et l'affiche dans le panneau Sortie lorsque la liste déroulante est modifiable, puis affiche « `CA` » lorsqu'elle ne l'est pas.

```
my_cb.dataProvider = [
  {label:"Alaska", data:"AK"},
  {label:"California", data:"CA"},
  {label:"Washington", data:"WA"}];
my_cb.editable = true;
my_cb.selectedIndex = 1;
trace('Editable value is "California": ' + my_cb.value);
my_cb.editable = false;
my_cb.selectedIndex = 1;
trace('Non-editable value is "CA": ' + my_cb.value);
```

CHAPITRE 9

Classes de liaison des données

9

Les classes de liaison des données fournissent les fonctionnalités d'exécution de la fonction de liaison des données dans Flash. Vous pouvez créer et configurer visuellement les liaisons de données dans l'environnement de programmation Flash via l'onglet Liaisons de l'inspecteur de composants. Vous pouvez également créer et configurer les liaisons par programmation à l'aide des classes du package mx.data.binding.

REMARQUE

Les classes de liaison des données sont prises en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Pour une présentation générale de la liaison des données et de la création visuelle de liaisons des données dans l'outil de programmation Flash, reportez-vous à *Liaison des données* dans *Utilisation de Flash.*

Disponibilité des classes de liaison des données à l'exécution

Pour pouvoir compiler votre fichier SWF, les fichiers SWC contenant le code binaire des classes de liaison des données et de services Web doivent être présents dans votre bibliothèque. Lorsque vous créez des liaisons de données dans Flash pendant la programmation, les classes de composants appropriées sont automatiquement ajoutées à la bibliothèque. Si vous travaillez avec les liaisons de données et les services Web à l'exécution, vous devez ajouter les classes dans la bibliothèque de votre fichier FLA. Vous pouvez obtenir ces fichiers SWC dans la bibliothèque commune Classes.

Pour ajouter les fichiers SWC à votre bibliothèque :

1. Assurez-vous que vos paramètres de publication sont bien définis sur ActionScript 2.0.
2. Sélectionnez la bibliothèque Classes (Fenêtre > Bibliothèques communes > Classes).
3. Ouvrez la bibliothèque de votre document en choisissant Fenêtre > Bibliothèque.
4. Faites glisser les fichiers SWC appropriés (DataBindingClasses et/ou WebServiceClasses) de la bibliothèque Classes vers celle de votre document.

Pour plus d'informations sur ces classes, reportez-vous à « Classe Binding », à la page 217 et à « Classes de service Web », à la page 1467.

Classes dans le package mx.data.binding

Le tableau suivant répertorie les classes du package mx.data.binding :

Classe	Description
Classe Binding	Crée une liaison entre deux points de fin.
Classe ComponentMixins	Ajoute une fonctionnalité de liaison aux composants.
Classe CustomFormatter	Classe de base pour la création de classes de mise en forme personnalisées.
CustomValidator, classe	Classe de base pour la création de classes de validation personnalisées.
Classe DataType	Offre un accès en lecture et en écriture aux champs de données d'une propriété de composant.
Classe EndPoint	Définit la source ou la destination d'une liaison.
Classe TypedValue	Contient une valeur de données et des informations concernant le type de données de la valeur.

Classe Binding

Nom de classe ActionScript mx.data.binding.Binding

La classe Binding définit une association entre deux points de fin, une source et une destination. Elle recherche les modifications apportées au point de fin source et copie les données modifiées dans le point de fin de destination chaque fois que la source change.

REMARQUE	La classe Binding est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Vous pouvez rédiger des liaisons personnalisées à l'aide de la classe Binding (et des classes de prise en charge), ou utiliser l'onglet Liaisons de l'inspecteur des composants. Pour rendre cette classe disponible à l'exécution, vous devez inclure les classes de liaison des données dans votre fichier FLA. Pour plus d'informations, voir « Disponibilité des classes de liaison des données à l'exécution », à la page 215.

Pour obtenir une vue d'ensemble des classes dans le package mx.data.binding, reportez-vous à « Classes dans le package mx.data.binding », à la page 216.

Méthodes de la classe Binding

Le tableau suivant présente les méthodes de la classe Binding.

Méthode	Description
`Binding.execute()`	Extrait les données du composant source, les met en forme et les affecte au composant de destination.

Constructeur de la classe Binding

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
new Binding(source, destination, [format], [isTwoWay])
```

Paramètres

source Point de fin source de la liaison. Ce paramètre est de type mx.data.binding.EndPoint, mais peut correspondre à tout objet ActionScript doté des champs de point de fin requis (voir « Classe EndPoint », à la page 230).

destination Point de fin de destination de la liaison. Ce paramètre est de type `mx.data.binding.EndPoint`, mais peut correspondre à tout objet ActionScript doté des champs de point de fin requis.

format Objet facultatif qui contient des informations sur le format. Les propriétés de l'objet doivent être les suivantes :

- `cls` Classe ActionScript 2.0 qui étend la classe mx.data.binding.DataAccessor.
- `settings` Objet dont les propriétés fournissent des paramètres facultatifs pour la classe de mise en forme spécifiée par `cls`.

isTwoWay Valeur booléenne facultative qui spécifie si le nouvel objet Binding est bidirectionnel (`true`) ou non (`false`). La valeur par défaut est `false`.

Renvoie

Aucune.

Description

Constructeur : crée un nouvel objet Binding. Vous pouvez lier des données à tout objet ActionScript qui dispose de propriétés et émet des événements dont, entre autres, des composants.

Un objet de liaison existe tant que le clip principal contient les composants source et destination. Par exemple, si un clip nommé A contient les composants X et Y et qu'il existe une liaison entre X et Y, la liaison est effective tant que le clip A existe.

REMARQUE

Il n'est pas nécessaire de conserver une référence au nouvel objet Binding. Dès sa création, l'objet Binding commence à écouter les événements changed émis par tous les points de fin. Toutefois, dans certains cas, vous pouvez choisir d'enregistrer une référence du nouvel objet Binding pour pouvoir ensuite appeler sa méthode `execute()` (voir `Binding.execute()`).

Exemple

Dans cet exemple, la propriété `text` d'un composant TextInput (`src_txt`) est liée à la propriété `text` d'un autre composant TextInput (`dest_txt`). Lorsque le champ de texte `src_txt` perd le focus (c'est-à-dire, lorsque l'événement `focusOut` est généré), la valeur de sa propriété `text` est copiée dans `dest_txt.text`.

```
import mx.data.binding.*;
var src = new EndPoint();
src.component = src_txt;
src.property = "text";
src.event = "focusOut";

var dest = new EndPoint();
dest.component = dest_txt;
dest.property = "text";

new Binding(src, dest);
```

Cet exemple décrit la création d'un objet Binding qui utilise une classe de mise en forme personnalisée. Pour plus d'informations, voir « Classe CustomFormatter », à la page 221.

```
import mx.data.binding.*;
var src = new EndPoint();
src.component = src_txt;
src.property = "text";
src.event = "focusOut";

var dest = new EndPoint();
dest.component = text_dest;
dest.property = "text";

new Binding(src, dest, {cls: mx.data.formatters.Custom, settings:
  {classname: "com.mycompany.SpecialFormatter"}});
```

Binding.execute()

Disponibilité

Flash Player 6.

Edition

Flash MX Professional 2004.

Utilisation

myBinding.execute([*reverse*])

Paramètres

reverse Valeur booléenne qui indique si la liaison doit également être exécutée de la destination vers la source (`true`), ou uniquement de la source vers la destination (`false`). Par défaut, cette valeur est définie sur `false`.

Renvoie

Une valeur `null` si la liaison est exécutée correctement. Dans le cas contraire, la méthode renvoie des chaînes de messages décrivant les erreurs qui ont empêché l'exécution de la liaison.

Description

Méthode qui recherche les données du composant source et les affecte au composant de destination. Si la liaison utilise une mise en forme, les données sont formatées avant leur affectation à la destination.

Cette méthode valide également les données et entraîne la production d'un événement `valid` ou `invalid` par les composants source et de destination. Les données sont affectées à l'événement de destination même s'il n'est pas valide, sauf si la destination est en lecture seule.

Lorsque le paramètre *reverse* est défini sur `true` et que la liaison est bidirectionnelle, la liaison est alors exécutée dans l'ordre inverse (de la destination vers la source).

Exemple

Le code suivant, associé à une occurrence de composant Button, exécute la liaison dans l'ordre inverse (du composant de destination vers le composant source) lorsque l'utilisateur clique sur le bouton.

```
on(click) {
  _root.myBinding.execute(true);
}
```

Classe CustomFormatter

Nom de classe ActionScript mx.data.binding.CustomFormatter

La classe CustomFormatter définit deux méthodes, `format()` et `unformat()`, qui permettent de transformer des valeurs de données d'un type spécifique en type String, et inversement. Par défaut, ces méthodes n'ont aucun rôle ; vous devez les implémenter dans une sous-classe de mx.data.binding. Pour rendre cette classe disponible à l'exécution, vous devez inclure les classes de liaison des données dans votre fichier FLA. Pour plus d'informations, voir « Disponibilité des classes de liaison des données à l'exécution », à la page 215.

REMARQUE	La classe CustomFormatter est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Pour créer votre propre mise en forme personnalisée, vous devez créer une sous-classe de CustomFormatter qui implémente les méthodes `format()` et `unformat()`. Vous pouvez ensuite affecter cette classe à une liaison entre des composants, soit en créant un objet Binding avec ActionScript (voir « Classe Binding », à la page 217), soit en utilisant l'onglet Liaisons de l'inspecteur de composants. Pour plus d'informations sur l'affectation d'une classe de mise en forme via l'inspecteur de composants, reportez-vous à la section traitant des objets de mise en forme (Formatters) des schémas dans *Utilisation de Flash*.

Vous pouvez également affecter une classe de mise en forme à une propriété de composant dans l'onglet Schéma de l'inspecteur de composants. Toutefois, dans ce cas, la mise en forme n'est utilisée que lorsque les données doivent prendre la forme d'une chaîne. A l'inverse, les mises en forme affectées via le panneau Liaisons, ou créées avec ActionScript, sont utilisées à chaque exécution de la liaison.

Pour obtenir un exemple d'écriture et d'affectation d'une mise en forme personnalisée à l'aide d'ActionScript, reportez-vous à « Exemple de mise en forme personnalisée », à la page 222.

Pour obtenir une vue d'ensemble des classes dans le package mx.data.binding, reportez-vous à « Classes dans le package mx.data.binding », à la page 216.

Exemple de mise en forme personnalisée

L'exemple suivant décrit la création d'une classe de mise en forme personnalisée, avant son application à une liaison entre deux composants via ActionScript. Dans cet exemple, la valeur actuelle d'un composant NumericStepper (sa propriété `value`) est liée à la valeur actuelle d'un composant TextInput (sa propriété `text`). La classe de mise en forme personnalisée met en forme la valeur numérique actuelle du composant NumericStepper (par exemple, 1, 2 ou 3) en utilisant le terme anglais équivalent (par exemple, « one », « two » ou « three ») avant de l'affecter au composant TextInput.

Pour créer et utiliser une mise en forme personnalisée :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier ActionScript.
2. Ajoutez le code suivant au fichier :

```
// NumberFormatter.as
class NumberFormatter extends mx.data.binding.CustomFormatter {
  // Met en forme un nombre, renvoie une chaîne
  function format(rawValue) {
    var returnValue;
    var strArray = new Array('one', 'two', 'three');
    var numArray = new Array(1, 2, 3);
    returnValue = 0;
    for (var i = 0; i<strArray.length; i++) {
      if (rawValue == numArray[i]) {
        returnValue = strArray[i];
        break;
      }
    }
    return returnValue;
  } // convertir une valeur mise en forme, renvoyer une valeur brute
  function unformat(formattedValue) {
    var returnValue;
    var strArray = new Array('one', 'two', 'three');
    var numArray = new Array(1, 2, 3);
    returnValue = "invalid";
    for (var i = 0; i<strArray.length; i++) {
      if (formattedValue == strArray[i]) {
        returnValue = numArray[i];
        break;
      }
    }
    return returnValue;
  }
}
```

3. Enregistrez le fichier sous le nom NumberFormatter.as.
4. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).

5. Faites glisser un composant TextInput du panneau Composants sur la scène et nommez-le **textInput**. Faites ensuite glisser un composant NumericStepper sur la scène et nommez-le **stepper**.
6. Ouvrez le scénario et sélectionnez la première image du Calque 1.
7. Dans le panneau Actions, ajoutez le code suivant :

```
import mx.data.binding.*;
var x:NumberFormatter;
var customBinding = new Binding({component:stepper, property:"value",
  event:"change"}, {component:textInput, property:"text",
  event:"enter,change"}, {cls:mx.data.formatters.Custom,
  settings:{classname:"NumberFormatter"}});
```

   La deuxième ligne de code (`var x:NumberFormatter`) garantit que le code binaire de votre classe de mise en forme personnalisée est inclus dans le fichier SWF compilé.
8. Choisissez Fenêtre > Bibliothèques communes > Classes pour ouvrir la bibliothèque Classes.
9. Sélectionnez Fenêtre > Bibliothèque pour ouvrir la bibliothèque de votre document.
10. Faites glisser DataBindingClasses de la bibliothèque Classes jusqu'à la bibliothèque de votre document.

    Ainsi, les classes d'exécution de liaison des données sont à la disposition de votre document.
11. Enregistrez le fichier FLA dans le dossier qui contient NumberFormatter.as.
12. Testez le fichier (Contrôle > Tester l'animation).

    Cliquez sur les boutons du composant NumericStepper et observez le contenu de la mise à jour du composant TextInput.

Méthodes de la classe CustomFormatter

Le tableau suivant présente les méthodes de la classe CustomFormatter.

Méthode	Description
`CustomFormatter.format()`	Convertit un type de données brutes en un nouvel objet.
`CustomFormatter.unformat()`	Convertit une chaîne, ou un autre type de données, en type de données brutes.

CustomFormatter.format()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Cette méthode est appelée automatiquement ; vous ne l'invoquez pas directement.

Paramètres

rawData Données à mettre en forme.

Renvoie

Une valeur mise en forme.

Description

Méthode qui convertit un type de données brutes en un nouvel objet.

Cette méthode n'est pas implémentée par défaut. Vous devez la définir dans votre sous-classe de mx.data.binding.CustomFormatter.

Pour plus d'informations, voir « Exemple de mise en forme personnalisée », à la page 222.

CustomFormatter.unformat()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Cette méthode est appelée automatiquement ; vous ne l'invoquez pas directement.

Paramètres

formattedData Données mises en forme à reconvertir en type de données brutes.

Renvoie

Une valeur non mise en forme.

Description

Méthode qui convertit une chaîne, ou un autre type de données, en type de données brutes. Cette transformation doit réaliser exactement la transformation inverse de `CustomFormatter.format()`.

Cette méthode n'est pas implémentée par défaut. Vous devez la définir dans votre sous-classe de mx.data.binding.CustomFormatter.

Pour plus d'informations, voir « Exemple de mise en forme personnalisée », à la page 222.

CustomValidator, classe

Nom de classe ActionScript mx.data.binding.CustomValidator

La classe CustomValidator permet d'effectuer une validation personnalisée d'un champ de données contenu dans un composant. Pour rendre cette classe disponible à l'exécution, vous devez inclure les classes de liaison des données dans votre fichier FLA. Pour plus d'informations, voir « Disponibilité des classes de liaison des données à l'exécution », à la page 215.

REMARQUE	La classe CustomValidator est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Pour créer une classe de validation personnalisée, créez tout d'abord une sous-classe de mx.data.binding.CustomValidator qui implémente une méthode appelée `validate()`. Une valeur est automatiquement transmise à cette méthode pour sa validation. Pour plus d'informations sur l'implémentation de cette méthode, consultez la section `CustomValidator.validate()`.

Affectez ensuite votre classe de validation personnalisée à un champ d'un composant via l'onglet Schéma de l'inspecteur des composants. Pour obtenir un exemple de création et d'utilisation d'une classe de validation personnalisée, consultez la section Exemple dans la description de `CustomValidator.validate()`.

Pour affecter une validation personnalisée :

1. Assurez-vous que vos paramètres de publication sont bien définis sur ActionScript 2.0.
2. Dans l'inspecteur des composants, sélectionnez l'onglet Schéma.
3. Sélectionnez le champ à valider, puis choisissez Personnalisé dans le menu déroulant Types de données.

4. Double-cliquez sur le champ Options de validation pour ouvrir la boîte de dialogue Paramètres de validation personnalisés.
5. Dans la zone de texte Classe ActionScript, entrez le nom de la classe de validation personnalisée que vous avez créée.

   Pour que la classe spécifiée soit incluse dans le fichier SWF publié, elle doit figurer dans le chemin de classe.

Pour une présentation générale des classes dans le package mx.data.binding, reportez-vous à « Classes dans le package mx.data.binding », à la page 216.

Méthodes de la classe CustomValidator

Le tableau suivant présente les méthodes de la classe CustomValidator.

Méthode	Description
`CustomValidator.validate()`	Effectue la validation des données.
`CustomValidator.validationError()`	Signale les erreurs de validation.

CustomValidator.validate()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Cette méthode est appelée automatiquement ; vous ne l'invoquez pas directement.

Paramètres

value Données à valider ; cette valeur peut être de n'importe quel type.

Renvoie

Aucune.

Description

Méthode ; appelée automatiquement pour valider les données contenues par le paramètre *value*. Vous devez implémenter cette méthode dans votre sous-classe de CustomValidator ; l'implémentation par défaut n'a aucune incidence.

Vous pouvez utiliser le code ActionScript de votre choix pour examiner et valider les données. Si les données ne sont pas valides, cette méthode doit appeler `this.validationError()` avec un message d'erreur approprié. Vous pouvez appeler `this.validationError()` à plusieurs reprises s'il existe plusieurs problèmes au niveau des données.

La méthode `validate()` pouvant être appelée de façon répétée, évitez d'ajouter du code long à exécuter. Votre implémentation de cette méthode ne doit vérifier que la validité des données, puis signaler les erreurs à l'aide de `CustomValidator.validationError()`. De même, votre implémentation ne doit prendre aucune mesure suite au test de validation (par exemple, alerter l'utilisateur final). En fait, vous devez créer des écouteurs pour les événements `valid` et `invalid`, puis alerter l'utilisateur final à partir de ces écouteurs d'événements (voir l'exemple suivant).

Exemple

La procédure suivante décrit la création et l'utilisation d'une classe de validation personnalisée. OddNumbersOnly.as, la méthode `validate()` de la classe CustomValidator, détermine comme non valide toute valeur qui n'est pas un nombre impair. La validation a lieu à chaque changement de la valeur d'un composant NumericStepper, en relation avec la propriété `text` d'un composant Label.

Pour créer et utiliser une classe de validation personnalisée :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier ActionScript.
2. Ajoutez le code suivant au fichier :

```
class OddNumbersOnly extends mx.data.binding.CustomValidator
{
  public function validate(value) {
    // Vérification du type Nombre de la valeur
    var n = Number(value);
    if (String(n) == "NaN") {
      this.validationError("'" + value + "' is not a number.");
      return;
    }
    // Vérification que le nombre est impair
    if (n % 2 == 0) {
      this.validationError("'" + value + "' is not an odd number.");
      return;
    }
    // Les données sont valides, aucune action requise,
    // renvoi uniquement
  }
}
```

3. Enregistrez le fichier sous OddNumbersOnly.as.

REMARQUE	Le nom du fichier AS doit correspondre au nom de la classe.

4. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
5. Ouvrez le panneau Composants.
6. Faites glisser un composant NumericStepper du panneau Composants sur la scène et appelez-le **stepper**.
7. Faites glisser un composant Label sur la scène et appelez-le **textLabel**.
8. Faites glisser un composant TextArea sur la scène et appelez-le **status**.
9. Sélectionnez le composant NumericStepper et ouvrez l'inspecteur des composants.
10. Ouvrez l'onglet Liaisons dans l'inspecteur des composants, puis cliquez sur le bouton Ajouter une liaison (+).
11. Sélectionnez la propriété Value (la seule) dans la boîte de dialogue Ajouter une liaison, puis cliquez sur OK.
12. Double-cliquez sur le champ Lié à pour ouvrir la boîte de dialogue connexe.
13. Dans la boîte de dialogue Lié à, sélectionnez le composant Label dans le panneau Chemin du composant, et sa propriété `text` dans le panneau Emplacement du schéma. Cliquez sur OK.
14. Sélectionnez le composant Label sur la scène et ouvrez l'onglet Schéma de l'inspecteur des composants.
15. Dans le panneau des attributs du schéma, sélectionnez Personnalisé dans le menu déroulant Types de données.
16. Double-cliquez sur le champ Validation options dans le panneau des attributs du schéma pour ouvrir la boîte de dialogue Paramètres de validation personnalisés.
17. Dans la zone de texte Classe ActionScript, tapez **OddNumbersOnly**, autrement dit le nom de la classe ActionScript créée préalablement. Cliquez sur OK.
18. Ouvrez le scénario et sélectionnez la première image du Calque 1.
19. Ouvrez le panneau Actions.

20. Ajoutez le code suivant au panneau Actions :

```
function dataIsInvalid(evt) {
  if (evt.property == "text") {
    status.text = evt.messages;
  }
}

function dataIsValid(evt) {
  if (evt.property == "text") {
    status.text = "OK";
  }
}

textLabel.addEventListener("valid", dataIsValid);
textLabel.addEventListener("invalid", dataIsInvalid);
```

21. Enregistrez le fichier FLA sous OddOnly.fla dans le dossier qui contient OddNumbersOnly.as.

22. Testez le fichier SWF (Contrôle > Tester l'animation).

Cliquez sur les flèches du composant NumericStepper afin de modifier sa valeur. Vous pouvez remarquer qu'un message apparaît dans le composant TextArea lorsque vous choisissez des nombres pairs et impairs.

CustomValidator.validationError()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`this.validationError(`*`errorMessage`*`)`

REMARQUE	Cette méthode peut être invoquée uniquement depuis une classe de validation personnalisée ; le mot clé `this` fait référence à l'objet CustomValidator actif.

Paramètres

`errorMessage` Chaîne qui contient le message d'erreur à signaler.

Renvoie

Aucune.

Description

Méthode ; appelée depuis la méthode `validate()` de votre sous-classe de CustomValidator pour signaler les erreurs de validation. Si vous n'appelez pas `validationError()`, un événement `valid` est généré lorsque `validate()` a fini de s'exécuter. Si vous appelez `validationError()` une ou plusieurs fois à partir de la méthode `validate()`, un événement `invalid` est généré après le renvoi de `validate()`.

Chaque message transmis à `validationError()` est disponible dans la propriété `messages` de l'objet événement transmis au gestionnaire d'événements `invalid`.

Exemple

Consultez la section Exemple de `CustomValidator.validate()`.

Classe EndPoint

Nom de classe ActionScript mx.data.binding.EndPoint

La classe EndPoint définit la source ou la destination d'une liaison. Les objets EndPoint définissent une constante, une propriété de composant ou un champ particulier d'une propriété de composant, à partir desquels vous pouvez obtenir des données, ou auxquels vous pouvez affecter des données. Ils peuvent également définir un événement ou une liste d'événements qu'un objet Binding attend. Lorsque l'événement spécifié se produit, la liaison est exécutée. Pour rendre cette classe disponible à l'exécution, vous devez intégrer les classes de liaison des données dans votre fichier FLA. Pour plus d'informations, voir « Disponibilité des classes de liaison des données à l'exécution », à la page 215.

REMARQUE	La classe EndPoint est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Lorsque vous créez une nouvelle liaison avec le constructeur de classe Binding, vous la transmettez à deux objets EndPoint : un pour la source et un pour la destination.

```
new mx.data.binding.Binding(srcEndPoint, destEndPoint);
```

Les objets EndPoint, `srcEndPoint` et `destEndPoint`, peuvent être définis comme suit :

```
var srcEndPoint = new mx.data.binding.EndPoint();
var destEndPoint = new mx.data.binding.EndPoint();
srcEndPoint.component = source_txt;
srcEndPoint.property = "text";
srcEndPoint.event = "focusOut";
destEndPoint.component = dest_txt;
destEndPoint.property = "text";
```

Le code précédent signifie « lorsque le champ de texte source perd le focus, copiez la valeur de sa propriété `text` dans la propriété `text` du champ de texte de destination ».

Vous pouvez également transmettre des objets ActionScript génériques au constructeur Binding, au lieu de transmettre explicitement des objets EndPoint construits. La seule condition veut que les objets définissent les propriétés EndPoint nécessaires, à savoir `component` et `property`. Le code suivant est similaire au code présenté précédemment.

```
var srcEndPoint = {component:source_txt, property:"text"};
var destEndPoint = {component:dest_txt, property:"text"};
new mx.data.binding.Binding(srcEndPoint, destEndPoint);
```

Pour une présentation générale des classes dans le package mx.data.binding, reportez-vous à « Classes dans le package mx.data.binding », à la page 216.

Propriétés de la classe EndPoint

Le tableau suivant présente les propriétés de la classe EndPoint.

Méthode	Description
`EndPoint.component`	Référence à une occurrence de composant.
`EndPoint.constant`	Valeur constante.
`EndPoint.event`	Nom d'un événement, ou tableau de noms d'événements, que le composant émet lorsque les données changent.
`EndPoint.location`	Emplacement d'un champ de données dans la propriété de l'occurrence du composant.
`EndPoint.property`	Nom d'une propriété de l'occurrence du composant spécifiée par `EndPoint.component`.

Constructeur de la classe EndPoint

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
new EndPoint()
```

Renvoie

Aucune.

Description

Constructeur qui crée un nouvel objet EndPoint.

Exemple

Cet exemple crée un objet EndPoint appelé `source_obj` et affecte des valeurs à ses propriétés `component` et `property` :

```
var source_obj = new mx.data.binding.EndPoint();
source_obj.component = myTextField;
source_obj.property = "text";
```

EndPoint.component

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`endPointObj``.component`

Description

Propriété : référence à une occurrence de composant.

Exemple

Cet exemple affecte une occurrence du composant List (`listBox1`) comme paramètre de composant d'un objet EndPoint.

```
var sourceEndPoint = new mx.data.binding.EndPoint();
sourceEndPoint.component = listBox1;
```

EndPoint.constant

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

endPoint_src.constant

Description

Propriété : constante affectée à un objet EndPoint. Cette propriété peut être affectée uniquement à des objets EndPoint qui sont la source (et non la destination) d'une liaison entre des composants. La valeur peut être tout type de données compatible avec la destination de la liaison. Si cette propriété est précisée, toutes les autres propriétés EndPoint de l'objet EndPoint spécifié sont ignorées.

Exemple

Dans cet exemple, la valeur de chaîne constante « hello » est affectée à la propriété constante d'un objet EndPoint :

```
var sourceEndPoint = new mx.data.binding.EndPoint();
sourceEndPoint.constant = "hello";
```

EndPoint.event

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

endPointObj.event

Description

Propriété qui spécifie le nom d'un événement, ou un tableau de noms d'événements, généré par le composant lorsque les données affectées à la propriété de liaison changent. Lorsque l'événement a lieu, la liaison est exécutée.

L'événement spécifié s'applique uniquement aux composants utilisés comme source d'une liaison, ou comme destination d'une liaison bidirectionnelle. Pour plus d'informations sur la création de liaisons bidirectionnelles, reportez-vous à « Classe Binding », à la page 217.

Exemple

Dans cet exemple, vous devez ajouter deux occurrences TextInput sur la scène. Un des champs de texte doit prendre le nom d'occurrence `first_textInput`, et l'autre `second_textInput`. Vous devez également ajouter le composant DataBindingClasses à la bibliothèque.

```
import mx.data.binding.Binding;
var src:Object = {component:first_textInput, property:"text",
  event:["enter", "focusOut"]};
var dst:Object = {component:second_textInput, property:"text"};
var my_binding:Binding = new Binding(src, dst);

first_textInput.addEventListener("enter", doEnter);
function doEnter(eventObject) {}
```

EndPoint.location

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

endPointObj.location

Description

Propriété qui spécifie l'emplacement d'un champ de données dans la propriété de l'occurrence du composant. Un emplacement peut être spécifié de quatre manières : sous forme de chaîne contenant une expression XPath ou un chemin ActionScript, sous forme de tableau de chaînes ou sous forme d'objet.

Les expressions XPath ne peuvent être utilisées que lorsque les données sont un objet XML. (Voir l'exemple 1 ci-après). Pour obtenir la liste des expressions XPath prises en charge, reportez-vous à *Ajout de liaisons à l'aide d'expressions de chemin* dans *Utilisation de Flash*.

Pour les objets XML et ActionScript, vous pouvez également spécifier une chaîne qui contient un chemin ActionScript. Un tel chemin contient les noms des champs séparés par des points (par exemple, `"a.b.c"`).

Vous pouvez également spécifier un tableau de chaînes comme emplacement. Chaque chaîne du tableau analyse un autre niveau d'imbrication. Vous pouvez utiliser cette technique avec des données XML et ActionScript (Voir l'exemple 2 ci-après). Lorsqu'un tableau de chaînes est utilisé avec des données ActionScript, cela équivaut à utiliser ActionScript ; c'est-à-dire que le tableau `["a","b","c"]` équivaut à `"a.b.c"`.

Si vous spécifiez un objet comme emplacement, il doit définir deux propriétés : `path` et `indices`. La propriété `path` est un tableau de chaînes, comme indiqué précédemment, sauf qu'une ou plusieurs des chaînes spécifiées peuvent être le symbole spécial `"[n]"`. Pour chaque occurrence de ce symbole dans `path`, un élément d'index correspondant doit exister dans `indices`. Lorsque le chemin est évalué, les index sont utilisés pour l'indexation dans les tableaux. L'élément d'index peut être tout objet EndPoint. Ce type d'emplacement ne peut pas être appliqué aux données XML mais exclusivement aux données ActionScript (voir l'exemple 3 ci-après).

Exemple

Exemple 1 : Cet exemple utilise une expression XPath pour désigner l'emplacement d'un nœud appelé `zip` dans un objet XML :

```
var sourceEndPoint = new mx.databinding.EndPoint();
var sourcObj = new Object();
sourceObj.xml = new XML("<zip>94103</zip>");
sourceEndPoint.component = sourceObj;
sourceEndPoint.property = "xml";
sourceEndPoint.location = "/zip";
```

Exemple 2 : Cet exemple utilise un tableau de chaînes pour effectuer une analyse poussée d'une propriété de clip imbriquée :

```
var sourceEndPoint = new mx.data.binding.EndPoint();
// Suppose que movieClip1.ball.position existe.
sourceEndPoint.component = movieClip1;
sourceEndPoint.property = "ball";
// Accès à movieClip1.ball.position.x.
sourceEndPoint.location = ["position","x"];
```

Exemple 3 : Cet exemple indique comment utiliser un objet pour désigner l'emplacement d'un champ de données dans une structure complexe :

```
var city = new Object();
city.theaters = [{theater: "t1", movies: [{name: "Good,Bad,Ugly"},
  {name:"Matrix Reloaded"}]}, {theater: "t2", movies: [{name: "Gladiator"},
  {name: "Catch me if you can"}]}];
var srcEndPoint = new EndPoint();
srcEndPoint.component = city;
srcEndPoint.property = "theaters";
srcEndPoint.location = {path: ["[n]","movies","[n]","name"], indices:
  [{constant:0},{constant:0}]};
```

EndPoint.property

Disponibilité

Flash Player 6 (6.0.79.0)

Edition

Flash MX Professional 2004.

Utilisation

endPointObj.property

Description

Propriété : désigne un nom de propriété de l'occurrence de composant spécifiée par `EndPoint.component` qui contient les données susceptibles d'être liées.

REMARQUE

`EndPoint.component` et `EndPoint.property` doivent être associés pour former une combinaison objet/propriété ActionScript valide.

Exemple

Cet exemple lie la propriété `text` d'un composant TextInput (`text_1`) à la même propriété d'un autre composant TextInput (`text_2`).

```
var sourceEndPoint = {component:text_1, property:"text"};
var destEndPoint = {component:text_2, property:"text"};
new Binding(sourceEndPoint, destEndPoint);
```

Classe ComponentMixins

Nom de classe ActionScript mx.data.binding.ComponentMixins

La classe ComponentMixins définit les propriétés et les méthodes automatiquement ajoutées à un objet correspondant à la source ou à la destination d'une liaison, ou encore à un composant qui est la cible d'un appel à la méthode `ComponentMixins.initComponent()`. Ces propriétés et méthodes n'ont aucune incidence sur le fonctionnement du composant, mais ajoutent des fonctionnalités utiles pour la liaison des données. Pour rendre cette classe disponible à l'exécution, vous devez intégrer les classes de liaison des données dans votre fichier FLA. Pour plus d'informations, voir « Disponibilité des classes de liaison des données à l'exécution », à la page 215.

REMARQUE

La classe ComponentMixins est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Pour obtenir une vue d'ensemble des classes dans le package mx.data.binding, reportez-vous à « Classes dans le package mx.data.binding », à la page 216.

Méthodes de la classe ComponentMixins

Le tableau suivant présente les méthodes de la classe ComponentMixins.

Méthode	Description
`ComponentMixins.getField()`	Renvoie un objet permettant d'obtenir et de définir la valeur d'un champ à un emplacement spécifique dans une propriété de composant.
`ComponentMixins.initComponent()`	Ajoute les méthodes ComponentMixin à un composant.
`ComponentMixins.refreshDestinations()`	Exécute toutes les liaisons qui ont cet objet comme objet Endpoint source.
`ComponentMixins.refreshFromSources()`	Exécute toutes les liaisons qui ont ce composant comme objet Endpoint de destination.
`ComponentMixins.validateProperty()`	Vérifie la validité des données de la propriété indiquée.

ComponentMixins.getField()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

componentInstance.getField(*propertyName*, [*location*])

Paramètres

propertyName Chaîne contenant le nom d'une propriété du composant spécifié.

location Paramètre facultatif qui indique l'emplacement d'un champ au sein de la propriété du composant. Ceci est utile si la propriété du composant spécifiée par *propertyName* est une structure de données complexe et si vous êtes intéressé par un champ de cette structure. La propriété *location* peut prendre trois formes :

- Une chaîne contenant une expression XPath. Cette forme est valide uniquement pour les structures de données XML. Pour obtenir la liste des expressions XPath prises en charge, reportez-vous à *Ajout de liaisons à l'aide d'expressions de chemin* dans *Utilisation de Flash*.

- Une chaîne contenant des noms de champ séparés par des points, par exemple `"a.b.c"`. Cette forme est autorisée pour toutes les données complexes (ActionScript ou XML).
- Un tableau de chaînes dans lequel chaque chaîne est un nom de champ, par exemple `["a", "b", "c"]`. Cette forme est autorisée pour toutes les données complexes (ActionScript ou XML).

Renvoie

Un objet DataType.

Description

Méthode qui renvoie un objet DataType dont vous pouvez utiliser les méthodes pour obtenir ou définir la valeur de données dans la propriété d'un composant à l'emplacement de champ spécifié. Pour plus d'informations, voir « Classe DataType », à la page 244.

Exemple

Cet exemple utilise la méthode `DataType.setAsString()` pour définir la valeur d'un champ situé dans la propriété d'un composant. Dans ce cas, la propriété (`results`) est une structure de données complexe.

```
import mx.data.binding.*;
var field : DataType = myComponent.getField("results", "po.address.name1");
field.setAsString("Teri Randall");
```

Voir aussi

`DataType.setAsString()`

ComponentMixins.initComponent()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`mx.data.binding.ComponentMixins.initComponent(`*`componentInstance`*`)`

Paramètres

`componentInstance` Référence à une occurrence de composant.

Renvoie

Aucune.

Description

Méthode (statique) ; ajoute toutes les méthodes ComponentMixins au composant spécifié par *componentInstance*. Cette méthode est appelée automatiquement pour tous les composants impliqués dans une liaison de données. Pour que les méthodes ComponentMixins puissent être utilisées par un composant non impliqué dans une liaison de données, vous devez les appeler explicitement pour le composant.

Exemple

Le code suivant met les méthodes ComponentMixins à la disposition d'un composant DataSet :

```
mx.data.binding.ComponentMixins.initComponent(_root.myDataSet);
```

ComponentMixins.refreshDestinations()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

componentInstance`.refreshDestinations()`

Paramètres

Aucun.

Renvoie

Aucune.

Description

Méthode ; exécute toutes les liaisons pour lesquelles *componentInstance* est l'objet EndPoint source. Cette méthode vous permet d'exécuter des liaisons dont les sources n'émettent pas d'événement "data changed" (données modifiées).

Exemple

L'exemple suivant exécute toutes les liaisons pour lesquelles l'occurrence du composant DataSet appelée `user_data` est l'objet EndPoint source :

```
user_data.refreshDestinations();
```

ComponentMixins.refreshFromSources()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

componentInstance.refreshFromSources()

Paramètres

Aucun.

Renvoie

Aucune.

Description

Méthode ; exécute toutes les liaisons pour lesquelles *componentInstance* est l'objet EndPoint de destination. Cette méthode vous permet d'exécuter des liaisons qui ont des sources constantes ou des sources n'émettant pas d'événement « data changed » (données modifiées).

Exemple

L'exemple suivant exécute toutes les liaisons pour lesquelles l'occurrence du composant ListBox appelée `cityList` est l'objet EndPoint de destination :

```
cityList.refreshFromSources();
```

ComponentMixins.validateProperty()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

componentInstance.validateProperty(*propertyName*)

Paramètres

propertyName Chaîne contenant le nom d'une propriété appartenant à *componentInstance*.

Renvoie

Un tableau ou la valeur `null`.

Description

Méthode ; détermine si les données contenues dans la chaîne *propertyName* sont valides en fonction des paramètres de schéma de la propriété. Les paramètres de schéma de la propriété sont ceux spécifiés sous l'onglet Schéma, dans l'inspecteur des composants.

La méthode renvoie la valeur `null` si les données sont valides. Dans le cas contraire, elle renvoie un tableau de messages d'erreur au format chaîne.

La validation ne s'applique qu'aux champs pour lesquels des informations de schéma sont disponibles. Si un champ est un objet contenant d'autres champs, chaque champ « enfant » est validé, de manière récursive. Chaque champ distribue un événement `valid` ou `invalid`, selon le cas. Pour chaque champ de données contenu par *propertyName*, cette méthode distribue des événements `valid` ou `invalid`, comme suit :

- Si la valeur du champ est `null` et *n'est pas* obligatoire, la méthode renvoie la valeur `null`. Aucun événement n'est généré.
- Si la valeur du champ est `null` et *est* obligatoire, une erreur est renvoyée et un événement `invalid` est généré.
- Si la valeur du champ n'est pas `null` et que le schéma du champ ne possède *pas* de validateur, la méthode renvoie la valeur `null` et aucun événement n'est généré.
- Si la valeur n'est pas `null` et que le schéma du champ *définit* un validateur, les données sont alors traitées par l'objet validateur. Si les données sont valides, un événement `valid` est généré et la valeur `null` est renvoyée ; dans le cas contraire, un événement `invalid` est généré et un tableau de chaînes d'erreur est renvoyé.

Exemple

L'exemple suivant indique comment utiliser la méthode `validateProperty()` pour s'assurer que la longueur du texte saisi par un utilisateur est valide. Vous déterminez la validité de la longueur en définissant les paramètres de validation des chaînes (paramètre Validation Options) du type de données String, dans l'onglet Schéma de l'inspecteur des composants. Si l'utilisateur saisit une chaîne d'une longueur non valide dans un champ de texte, les messages d'erreur renvoyés par la méthode `validateProperty()` s'affichent dans le panneau Sortie.

Pour valider le texte saisi par un utilisateur dans un composant TextInput :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser un composant TextInput du panneau Composants sur la scène et appelez-le **zipCode_txt**.
3. Sélectionnez le composant TextInput et, dans l'inspecteur des composants, cliquez sur l'onglet Schéma.
4. Dans le panneau Arborescence de schéma (panneau supérieur de l'onglet Schéma), sélectionnez la propriété `text`.
5. Dans le panneau des attributs du schéma (panneau inférieur de l'onglet Schéma), sélectionnez ZipCode dans le menu déroulant du paramètre Data Type.
6. Ouvrez le scénario s'il ne l'est pas déjà.
7. Cliquez sur la première image du calque 1 dans le scénario, puis ouvrez le panneau Actions (Fenêtre > Actions).
8. Ajoutez le code suivant au panneau Actions :

```
// Ajout des méthodes ComponentMixin au composant TextInput.
// Cette étape est requise uniquement si le composant
// n'est pas déjà impliqué dans une liaison de données,
// en tant que source ou destination.
mx.data.binding.ComponentMixins.initComponent(zipCode_txt);
// Définition de la fonction d'écouteur d'événements pour le composant :
validateResults = function (eventObj) {
  var errors:Array = eventObj.target.validateProperty("text");
  if (errors != null) {
    trace(errors);
  }
};
// Enregistrement de la fonction d'écouteur avec le composant :
zipCode_txt.addEventListener("enter", validateResults);
```

9. Choisissez Fenêtre > Bibliothèques communes > Classes pour ouvrir la bibliothèque Classes.
10. Sélectionnez Fenêtre > Bibliothèque pour ouvrir la bibliothèque de votre document.
11. Faites glisser DataBindingClasses de la bibliothèque Classes jusqu'à la bibliothèque de votre document.

    Cette étape est nécessaire pour que le fichier SWF puisse utiliser les classes d'exécution de liaison des données au moment de son exécution.
12. Testez le fichier SWF en choisissant Contrôle > Tester l'animation.

    Dans le composant TextInput de la scène, entrez un code postal américain non valide ; par exemple, un code qui contient des lettres ou moins de cinq chiffres. Vérifiez les messages d'erreur dans le panneau de sortie.

Classe DataType

Nom de classe ActionScript mx.data.binding.DataType

La classe DataType offre un accès en lecture et écriture aux champs de données d'une propriété de composant. Pour obtenir un objet DataType, vous devez appeler la méthode `ComponentMixins.getField()` sur un composant. Vous pouvez ensuite appeler les méthodes de l'objet DataType pour obtenir et définir la valeur du champ. Pour rendre cette classe disponible à l'exécution, vous devez intégrer les classes de liaison des données dans votre fichier FLA. Pour plus d'informations, voir « Disponibilité des classes de liaison des données à l'exécution », à la page 215.

REMARQUE	La classe DataType est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Si vous obtenez et définissez des valeurs de champ directement sur l'occurrence de composant au lieu d'utiliser des méthodes de classe DataType, les données sont fournies sous leur forme « brute ». Inversement, lorsque vous obtenez ou définissez des valeurs de champs à l'aide des méthodes de la classe DataType, ces valeurs sont traitées selon les paramètres de schéma du champ.

Par exemple, le code suivant obtient directement la valeur de la propriété d'un composant puis l'affecte à une variable. La variable `propVar` contient la valeur « brute » correspondant à la valeur actuelle de la propriété `propName`.

```
var propVar = myComponent.propName;
```

L'exemple suivant obtient la valeur de cette même propriété via la méthode `DataType.getAsString()`. Dans ce cas, la valeur affectée à `stringVar` est la valeur de `propName` suite à son traitement selon ses paramètres de schéma, puis à son renvoi sous forme de chaîne.

```
var dataTypeObj:mx.data.binding.DataType =
  myComponent.getField("propName");
var stringVar: String = dataTypeObj.getAsString();
```

Pour plus d'informations sur la définition des paramètres de schéma d'un champ, reportez-vous à *Utilisation des schémas dans l'onglet Schéma* dans *Utilisation de Flash*.

Vous pouvez également utiliser les méthodes de la classe DataType pour obtenir ou définir des champs dans différents types de données. La classe DataType convertit automatiquement les données brutes dans le type demandé, lorsque cela est possible. Par exemple, dans l'exemple de code précédent, les données extraites sont converties au format String (Chaîne), même si les données brutes sont d'un type différent.

La méthode `ComponentMixins.getField()` peut être utilisée pour les composants inclus dans une liaison de données (en tant que source, destination ou index) ou ayant été initialisés avec `ComponentMixins.initComponent()`. Pour plus d'informations, voir « Classe ComponentMixins », à la page 237.

Pour obtenir une vue d'ensemble des classes dans le package mx.data.binding, reportez-vous à « Classes dans le package mx.data.binding », à la page 216.

Méthodes de la classe DataType

Le tableau suivant présente les méthodes de la classe DataType.

Méthode	Description
`DataType.getAnyTypedValue()`	Extrait la valeur active du champ.
`DataType.getAsBoolean()`	Extrait la valeur active du champ sous forme de valeur booléenne.
`DataType.getAsNumber()`	Extrait la valeur active du champ sous forme de nombre.
`DataType.getAsString()`	Extrait la valeur active du champ sous forme de chaîne.
`DataType.getTypedValue()`	Extrait la valeur active du champ sous la forme du type de données demandé.
`DataType.setAnyTypedValue()`	Définit une nouvelle valeur dans le champ.
`DataType.setAsBoolean()`	Définit le champ sur la nouvelle valeur, donnée sous forme booléenne.
`DataType.setAsNumber()`	Définit le champ sur la nouvelle valeur, donnée sous forme de nombre.
`DataType.setAsString()`	Définit le champ sur la nouvelle valeur, donnée sous forme de chaîne.
`DataType.setTypedValue()`	Définit une nouvelle valeur dans le champ.

Propriétés de la classe DataType

Le tableau suivant présente les propriétés de la classe DataType.

Propriété	Description
`DataType.encoder`	Fournit une référence à l'objet Encoder associé à ce champ.
`DataType.formatter`	Fournit une référence à l'objet de mise en forme (Formatter) associé à ce champ.
`DataType.kind`	Fournit une référence à l'objet Kind associé à ce champ.

DataType.encoder

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`dataTypeObject``.encoder`

Description

Propriété qui fournit une référence à l'objet Encoder associé à ce champ, s'il existe.
Vous pouvez utiliser cette propriété pour accéder aux propriétés et aux méthodes définies par l'encodeur spécifique appliqué au champ dans l'onglet Schéma de l'inspecteur des composants.

Si aucun encodeur n'a été appliqué au champ concerné, la propriété renvoie la valeur `undefined`.

Pour plus d'informations sur les encodeurs fournis avec Flash, reportez-vous à *Encodeurs de schémas* dans *Utilisation de Flash*.

Exemple

Dans l'exemple suivant, nous supposons que le champ auquel l'utilisateur accède (`valide`) utilise l'encodeur Boolean (`mx.data.encoders.Bool`). Cet encodeur est fourni avec Flash et contient une propriété appelée `trueStrings` qui spécifie les chaînes devant être interprétées en tant que valeurs `true`. Le code suivant définit la propriété `trueStrings` de l'encodeur d'un champ afin qu'elle corresponde aux chaînes « yes » et « oui ».

```
var myField:mx.data.binding.DataType = dataSet.getField("isValid");
myField.encoder.trueStrings = "Yes,Oui";
```

DataType.formatter

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dataTypeObject.formatter

Description

Propriété qui fournit une référence à l'objet de mise en forme (Formatter) associé à ce champ, s'il existe. Vous pouvez utiliser cette propriété pour accéder aux propriétés et aux méthodes définies par l'objet de mise en forme appliqué au champ dans l'onglet Schéma de l'inspecteur des composants.

Si aucun objet de mise en forme n'a été appliqué au champ concerné, la propriété renvoie la valeur `undefined`.

Pour plus d'informations sur les objets de mise en forme fournis avec Flash, reportez-vous à la section traitant des objets de mise en forme (Formatters) des schémas dans *Utilisation de Flash*.

Exemple

Dans cet exemple, nous supposons que le champ auquel l'utilisateur accède utilise l'objet Number Formatter (`mx.data.formatters.NumberFormatter`) inclus dans Flash. Cet objet de mise en forme contient une propriété appelée `precision` qui indique le nombre de chiffres à afficher après la virgule. Ce code définit la propriété `precision` à deux chiffres après la virgule pour un champ utilisant l'objet NumberFormatter.

```
var myField:DataType = dataGrid.getField("currentBalance");
myField.formatter.precision = 2;
```

DataType.getAnyTypedValue()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dataTypeObject.getAnyTypedValue(*suggestedTypes*)

Paramètres

suggestedTypes Tableau de chaînes spécifiant, dans l'ordre décroissant, les types de donnés à utiliser en priorité pour le champ.

Renvoie

La valeur actuelle du champ, sous la forme d'un des types de données spécifiés dans le tableau *suggestedTypes*.

Description

Méthode ; extrait la valeur active du champ et la traite à l'aide des informations de schéma du champ. Si le champ est en mesure de fournir une valeur comme premier type de données spécifié dans le tableau *suggestedTypes*, la méthode renvoie la valeur sous la forme de ce type de données. Dans le cas contraire, la méthode tente d'extraire la valeur du champ en utilisant le deuxième type de données spécifié dans le tableau *suggestedTypes*, et ainsi de suite.

Si vous spécifiez `null` comme l'un des éléments dans le tableau *suggestedTypes*, la méthode renvoie la valeur du champ dans le type de données spécifié dans l'onglet Schéma de l'inspecteur de composants. La spécification de `null` entraînant systématiquement le renvoi d'une valeur, utilisez `null` uniquement à la fin du tableau.

Lorsqu'une valeur ne peut pas être renvoyée sous la forme de l'un des types suggérés, elle est renvoyée sous la forme du type spécifié dans l'onglet Schéma.

Exemple

Cet exemple tente d'obtenir la valeur d'un champ (`productInfo.available`) dans la propriété `results` d'un composant XMLConnector, tout d'abord sous forme de nombre, ou en cas d'échec, sous forme de chaîne.

```
import mx.data.binding.DataType;
import mx.data.binding.TypedValue;
var f: DataType = myXmlConnector.getField("results",
  "productInfo.available");
var b: TypedValue = f.getAnyTypedValue(["Number", "String"]);
```

Voir aussi

`ComponentMixins.getField()`

DataType.getAsBoolean()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dataTypeObject.getAsBoolean()

Paramètres

Aucun.

Renvoie

Une valeur booléenne.

Description

Méthode qui extrait la valeur actuelle du champ et la convertit en valeur booléenne, si nécessaire.

Exemple

Dans cet exemple, un champ appelé `propName` appartenant au composant `myComponent` est récupéré sous forme de valeur booléenne, puis affecté à une variable :

```
var dataTypeObj:mx.data.binding.DataType =
  myComponent.getField("propName");
var propValue:Boolean = dataTypeObj.getAsBoolean();
```

DataType.getAsNumber()

Disponibilité

Flash Player 6.

Edition

Flash MX Professional 2004.

Utilisation

dataTypeObject.getAsNumber()

Paramètres

Aucun.

Renvoie

Un nombre.

Description

Méthode qui extrait la valeur actuelle du champ et la convertit en nombre, si nécessaire.

Exemple

Dans cet exemple, un champ appelé `propName` appartenant au composant `myComponent` est récupéré sous forme de nombre, puis affecté à une variable :

```
var dataTypeObj:mx.data.binding.DataType =
  myComponent.getField("propName");
var propValue:Number = dataTypeObj.getAsNumber();
```

Voir aussi

`DataType.getAnyTypedValue()`

DataType.getAsString()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`dataTypeObject``.getAsString()`

Paramètres

Aucun.

Renvoie

Une chaîne.

Description

Méthode qui extrait la valeur actuelle du champ et la convertit sous forme de chaîne, si nécessaire.

Exemple

Dans cet exemple, une propriété appelée `propName` appartenant au composant `myComponent` est récupérée sous forme de chaîne, puis affectée à une variable :

```
var dataTypeObj:mx.data.binding.DataType =
  myComponent.getField("propName");
var propValue:String = dataTypeObj.getAsString();
```

Voir aussi

`DataType.getAnyTypedValue()`

DataType.getTypedValue()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`dataTypeObject``.getTypedValue(`*`requestedType`*`)`

Paramètres

`requestedType` Chaîne contenant le nom d'un type de données, ou `null`.

Renvoie

Un objet TypedValue (voir « Classe TypedValue », à la page 257).

Description

Méthode qui renvoie la valeur du champ sous la forme spécifiée, si le champ peut fournir sa valeur sous cette forme. Si le champ n'est pas en mesure de fournir sa valeur sous la forme demandée, la méthode renvoie la valeur `null`.

Si le paramètre *`requestedType`* a la valeur `null`, la méthode renvoie la valeur du champ dans son type par défaut.

Exemple

L'exemple suivant renvoie la valeur du champ convertie en type de données booléen et stockée dans la variable `bool`.

```
var bool:TypedValue = field.getTypedValue("Boolean");
```

DataType.kind

Disponibilité
Flash Player 6 (6.0.79.0).

Edition
Flash MX Professional 2004.

Utilisation
dataTypeObject.kind

Description
Propriété qui fournit une référence à l'objet Kind associé à ce champ. Vous pouvez utiliser cette propriété pour accéder aux propriétés et aux méthodes de l'objet Kind.

DataType.setAnyTypedValue()

Disponibilité
Flash Player 6 (6.0.79.0).

Edition
Flash MX Professional 2004.

Utilisation
dataTypeObject.setAnyTypedValue(*newTypedValue*)

Paramètres
newTypedValue Valeur d'un objet TypedValue à définir dans le champ. Pour plus d'informations, voir « Classe TypedValue », à la page 257.

Renvoie
Un tableau de chaînes décrivant les erreurs qui se sont produites lors de la définition de la nouvelle valeur. Des erreurs peuvent se produire dans les cas de figure suivants :

- Les données fournies ne peuvent pas être converties dans le type de données de ce champ (par exemple, la chaîne "abc" ne peut pas être convertie au format numérique).
- Les données sont d'un type acceptable mais ne répondent pas aux critères de validation du champ.
- Le champ est en lecture seule.

REMARQUE	Le texte du message d'erreur varie en fonction du type de données, des mises en forme et des encodeurs définis dans le schéma du champ.

Description

Méthode qui définit une nouvelle valeur dans le champ, à l'aide de ses informations de schéma, pour traiter le champ.

Cette méthode fonctionne en appelant d'abord la méthode `DataType.setTypedValue()` pour définir la valeur. Si cette opération échoue, la méthode vérifie si l'objet de destination accepte des données au format String, Boolean ou Number. Si c'est le cas, elle tente ensuite d'utiliser les fonctions de conversion ActionScript correspondantes.

Exemple

Cet exemple crée un objet TypedValue (valeur booléenne), puis affecte cette valeur à un objet DataType appelé `field`. Les erreurs qui se produisent sont affectées au tableau `erreurs`.

```
import mx.data.binding.*;
var t:TypedValue = new TypedValue (true, "Boolean");
var errors: Array = field.setAnyTypedValue (t);
```

Voir aussi

`DataType.setTypedValue()`

DataType.setAsBoolean()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`dataTypeObject``.setAsBoolean(`*`newBooleanValue`*`)`

Paramètres

`newBooleanValue` Valeur booléenne.

Renvoie

Aucune.

Description

Méthode qui définit le champ sur la nouvelle valeur, donnée sous forme booléenne. La valeur est convertie au type de données approprié pour ce champ, puis stockée sous ce format.

Exemple

L'exemple suivant définit une variable appelée `bool` sur la valeur booléenne `true`. Elle définit ensuite la valeur référencée par `field` sur `true`.

```
var bool: Boolean = true;
field.setAsBoolean (bool);
```

DataType.setAsNumber()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`dataTypeObject``.setAsNumber(`*`newNumberValue`*`)`

Paramètres

`newNumberValue` Nombre.

Renvoie

Aucune.

Description

Méthode qui définit le champ sur la nouvelle valeur, donnée sous forme de nombre. La valeur est convertie au type de données approprié pour ce champ, puis stockée sous ce format.

Exemple

L'exemple suivant définit une variable appelée `num` sur la valeur numérique 32. Elle définit ensuite la valeur référencée par `field`, sur `num`.

```
var num: Number = 32;
field.setAsNumber (num);
```

DataType.setAsString()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dataTypeObject.setAsString(*newStringValue*)

Paramètres

newStringValue String.

Renvoie

Aucune.

Description

Méthode qui définit le champ sur la nouvelle valeur, donnée sous forme de chaîne. La valeur est convertie au type de données approprié pour ce champ, puis stockée sous ce format.

Exemple

L'exemple suivant définit la variable `stringVal` sur la chaîne « `The new value (La nouvelle valeur)` ». Il définit ensuite la valeur référencée par `field` sur la chaîne.

```
var stringVal: String = "The new value";
field.setAsString (stringVal);
```

DataType.setTypedValue()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dataTypeObject.setTypedValue(*newTypedValue*)

Paramètres

newTypedValue Valeur d'un objet TypedValue à définir dans le champ.

Pour plus d'informations sur les objets TypedValue, reportez-vous à « Classe TypedValue », à la page 257.

Renvoie

Un tableau de chaînes décrivant les erreurs qui se sont produites lors de la définition de la nouvelle valeur. Des erreurs peuvent se produire dans les cas de figure suivants :

- Les données fournies ne sont pas d'un type acceptable.
- Les données fournies ne peuvent pas être converties dans le type de données de ce champ (par exemple, la chaîne `"abc"` ne peut pas être convertie au format numérique).
- Les données sont d'un type acceptable mais ne répondent pas aux critères de validation du champ.
- Le champ est en lecture seule.

REMARQUE	Le texte du message d'erreur varie en fonction du type de données, des mises en forme et des encodeurs définis dans le schéma du champ.

Description

Méthode qui définit une nouvelle valeur dans le champ, à l'aide de ses informations de schéma, pour traiter le champ. Cette méthode agit de façon similaire à la méthode `DataType.setAnyTypedValue()`, sauf qu'elle ne tente pas de convertir les données en un type de données acceptable. Pour plus d'informations, voir `DataType.setAnyTypedValue()`.

Exemple

Cet exemple crée un objet TypedValue (valeur booléenne), puis affecte cette valeur à un objet DataType appelé `field (champ)`. Les erreurs qui se produisent sont affectées au tableau `erreurs`.

```
import mx.data.binding.*;
var bool:TypedValue = new TypedValue (true, "Boolean");
var errors: Array = field.setTypedValue (bool);
```

Voir aussi

`DataType.setTypedValue()`

Classe TypedValue

Nom de classe ActionScript mx.data.binding.TypedValue

Cet objet contient une valeur de données et des informations concernant son type. Les objets TypedValue sont utilisés en tant que paramètres et renvoyés par plusieurs méthodes de la classe DataType. Les informations sur le type de données de l'objet TypedValue permettent aux objets DataType de déterminer quand et comment ils doivent effectuer la conversion du type de données. Pour rendre cette classe disponible à l'exécution, vous devez intégrer les classes de liaison des données dans votre fichier FLA. Pour plus d'informations, voir « Disponibilité des classes de liaison des données à l'exécution », à la page 215.

REMARQUE	La classe TypedValue est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Pour obtenir une vue d'ensemble des classes dans le package mx.data.binding, reportez-vous à « Classes dans le package mx.data.binding », à la page 216.

Propriétés de la classe TypedValue

Le tableau suivant présente les propriétés de la classe TypedValue.

Propriété	Description
`TypedValue.type`	Contient le schéma associé à la valeur de l'objet TypedValue.
`TypedValue.typeName`	Noms du type de données de la valeur de l'objet TypedValue.
`TypedValue.value`	Contient la valeur de données de l'objet TypedValue.

Constructeur de la classe TypedValue

Disponibilité

Flash Player 6 (6.0.79.0).

Utilisation

```
new mx.data.binding.TypedValue(value, typeName, [type])
```

Paramètres

value Valeur de données de tout type.

typeName Chaîne contenant le nom du type de données de la valeur.

type Objet Schema facultatif décrivant de façon détaillée le schéma des données. Ce champ est requis uniquement dans certaines circonstances, par exemple lors de la définition des données dans la propriété `dataProvider` d'un composant DataSet.

Description

Constructeur qui crée un nouvel objet TypedValue.

TypedValue.type

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
typedValueObject.type
```

Description

Propriété : contient le schéma associé à la valeur de l'objet TypedValue.

Exemple

Cet exemple affiche la valeur `null` dans le panneau Sortie :

```
var t: TypedValue = new TypedValue (true, "Boolean", null);
trace(t.type);
```

TypedValue.typeName

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

typedValueObject.typeName

Description

Propriété qui contient le nom du type de données de la valeur de l'objet TypedValue.

Exemple

Cet exemple affiche `Boolean` dans le panneau Sortie :

```
var t: TypedValue = new TypedValue (true, "Boolean", null);
trace(t.typeName);
```

TypedValue.value

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

typedValueObject.value

Description

Propriété qui contient la valeur de données de l'objet TypedValue.

Exemple

Cet exemple affiche la valeur `true` dans le panneau Sortie :

```
var t: TypedValue = new TypedValue (true, "Boolean", null);
trace(t.value);
```

CHAPITRE 10

Composant DataGrid

10

Ce composant vous permet de créer de puissantes applications et des affichages de données exhaustifs. Vous pouvez l'utiliser pour instancier un jeu d'enregistrements (extrait d'une requête de base de données dans Adobe ColdFusion, Java ou.Net) à l'aide d'Adobe Flash Remoting et l'afficher en colonnes. Vous pouvez également utiliser les données d'un ensemble ou d'un tableau pour remplir un composant DataGrid.

REMARQUE

Un composant DataGrid est pris en charge pour ActionScript 2.0 et ActionScript 3.0. Ce document présente le composant de la version 2. Si vous utilisez celui de la version 3, reportez-vous à « Utilisation du composant DataGrid » dans *Utilisation des composants ActionScript 3.0*.

Vous pouvez redimensionner et personnaliser des caractéristiques telles que la police, la couleur et les bordures des colonnes d'une grille. Vous pouvez utiliser un clip personnalisé en tant qu'objet CellRenderer pour toute colonne d'une grille. (Un objet Cell Renderer affiche le contenu d'une cellule.) Vous pouvez utiliser les barres de défilement pour faire défiler les données figurant dans une grille. Vous pouvez également désactiver les barres de défilement et utiliser les méthodes DataGrid pour créer un affichage de style mode Page. Pour plus d'informations sur la personnalisation, reportez-vous à « Classe DataGridColumn », à la page 315.

Lorsque vous ajoutez le composant DataGrid à une application, vous pouvez utiliser le panneau Accessibilité pour le rendre accessible aux logiciels de lecture d'écran. Commencez par ajouter la ligne de code suivante pour activer l'accessibilité du composant DataGrid :

```
mx.accessibility.DataGridAccImpl.enableAccessibility();
```

Quel que soit le nombre d'occurrences d'un composant, l'activation de son accessibilité ne se fait qu'une fois. Pour plus d'informations, reportez-vous à *Création de contenu accessible* dans *Utilisation de Flash*.

Interaction avec le composant DataGrid

La souris et le clavier permettent d'interagir avec un composant DataGrid.

Si `DataGrid.sortableColumns` et `DataGridColumn.sortOnHeaderRelease` sont tous deux définis sur `true`, l'utilisateur peut trier la grille en fonction des valeurs des cellules d'une colonne en cliquant sur l'en-tête de cette dernière.

Si `DataGrid.resizableColumns` est défini sur `true`, l'utilisateur peut redimensionner les colonnes en cliquant sur leur zone de séparation.

Si l'utilisateur clique sur une cellule modifiable, cette cellule reçoit le focus. S'il clique sur une cellule non modifiable, cela n'a aucun impact sur le focus. Une cellule est modifiable si ses propriétés `DataGrid.editable` et `DataGridColumn.editable` ont la valeur `true`.

Lorsqu'une occurrence de DataGrid a le focus (l'utilisateur a cliqué ou utilisé la tabulation), les touches suivantes permettent de la contrôler :

Touche	Description
Flèche vers le bas	Lorsque la cellule fait l'objet d'une modification, le point d'insertion se positionne à la fin du texte de la cellule. Si la cellule n'est pas modifiable, la flèche vers le bas gère la sélection de la même façon que le composant List.
Flèche vers le haut	Lorsque la cellule fait l'objet d'une modification, le point d'insertion se positionne au début du texte de la cellule. Si la cellule n'est pas modifiable, la flèche vers le haut gère la sélection de la même façon que le composant List.
Flèche droite	Lorsque la cellule fait l'objet d'une modification, le point d'insertion se déplace d'un caractère vers la droite. Si la cellule n'est pas modifiable, cette action n'a aucune incidence.
Flèche gauche	Lorsque la cellule fait l'objet d'une modification, le point d'insertion se déplace d'un caractère vers la gauche. Si la cellule n'est pas modifiable, cette action n'a aucune incidence.
Retour/Entrée/ Maj+Entrée	Lorsque la cellule est modifiable, la modification est validée et le point d'insertion se place dans la cellule de la ligne suivante (vers le haut ou vers le bas, en fonction du sens de basculement) dans la même colonne.
Maj+Tab/Tab	Place le focus sur l'élément précédent. Lorsque l'utilisateur appuie sur la touche de tabulation, le focus passe de la dernière colonne de la grille à la première colonne de la ligne suivante. Lorsque l'utilisateur appuie sur Maj+Tab, l'ordre est inversé. La totalité du texte de la cellule qui a le focus est sélectionnée.

Utilisation du composant DataGrid

Le composant DataGrid peut servir de base à de nombreux types d'applications de données. Vous pouvez aisément afficher une vue tabulaire d'une requête de base de données (ou autre), puis utiliser les fonctionnalités du composant CellRenderer pour créer des éléments d'interface utilisateur plus élaborés et modifiables. Voici des exemples concrets d'utilisation du composant DataGrid :

- Client de messagerie Web
- Pages de résultats de recherches
- Tableurs (applications de calculs de crédits et de formulaires de déclaration d'impôt)

Présentation de la conception du composant DataGrid

Le composant DataGrid étend les fonctions du Composant List. Lors du développement d'une application avec le composant DataGrid, une bonne connaissance de la conception de la classe List sous-jacente vous sera très utile. Voici quelques hypothèses et éléments requis utilisés par Adobe lors du développement de la classe List :

- Petite, rapide et simple.

  Ne jamais compliquer les choses plus que nécessaire. Voilà la première consigne à appliquer. La majorité des éléments requis énumérés ci-dessous respectent cette directive.
- Hauteurs de lignes des listes uniformes.

  Toutes les lignes doivent être de même hauteur. Celle-ci peut être définie à la programmation ou au moment de l'exécution.
- Les listes peuvent comporter des milliers d'enregistrements.
- Les listes ne mesurent pas le texte.

  Cela entraîne un problème de défilement horizontal pour les composants List et Tree. Pour plus d'informations, reportez-vous à « Fonctionnement du composant List », à la page 793. Le composant DataGrid prend toutefois en charge `"auto"` comme valeur de `hScrollPolicy`, car il mesure les colonnes (de même largeur par élément) et non le texte.

  Le fait que les listes ne mesurent pas le texte explique l'uniformité des hauteurs de lignes. Le dimensionnement individuel de chaque ligne en fonction du texte demanderait un très grand nombre de mesures. Par exemple, pour afficher avec précision les barres de défilement d'une liste de lignes aux hauteurs non uniformes, vous devrez d'abord mesurer chaque ligne.

- Les listes s'exécutent d'autant moins bien que le nombre de lignes visibles est important.

  Si elles peuvent contenir 5000 enregistrements, elles ne peuvent pas tous les afficher simultanément. Plus le nombre de lignes visibles sur la scène est important (défini par la propriété `rowCount`), plus la liste doit travailler au rendu. Lorsque cela est possible, limiter le nombre de lignes visibles est la meilleure solution.
- Les listes ne sont pas des tableaux.

  Les composants DataGrid sont utilisés pour afficher de nombreux enregistrements. Ils ne sont pas conçus pour afficher la totalité des informations, mais pour en présenter suffisamment pour que l'utilisateur puisse ensuite les développer. L'affichage des messages dans Microsoft Outlook en est un bon exemple. Il n'est pas nécessaire de lire la totalité du message dans la grille. La lecture serait très difficile et le client de messagerie aurait bien du mal à s'exécuter. Outlook présente seulement suffisamment d'informations pour que l'utilisateur en demande la totalité s'il est intéressé.

Présentation du composant DataGrid : affichage et modèle de données

A la base, le composant DataGrid est constitué d'un modèle de données et d'un affichage de présentation des données. Le modèle de données comprend trois parties :

- DataProvider (Fournisseur de données)

  Il s'agit de la liste des éléments qui doivent remplir la grille de données. Un tableau se trouvant dans la même image qu'un composant DataGrid se voit automatiquement attribuer des méthodes (de l'API DataProvider) qui permettent de manipuler des données et de diffuser les modifications dans plusieurs affichages. Tout objet qui implémente l'API DataProvider peut être affecté à la propriété `DataGrid.dataProvider` (notamment des jeux d'enregistrements, de données, etc.). Le code suivant crée un fournisseur de données appelé `myDP` :

  ```
  myDP = new Array({name:"Chris", price:"Priceless"}, {name:"Nigel",
    price:"Cheap"});
  ```
- Elément

  Il s'agit d'un objet ActionScript utilisé pour stocker les unités d'informations des cellules d'une colonne. Une grille de données est en fait une liste pouvant afficher plusieurs colonnes de données. Une liste fonctionne de la même façon qu'un tableau ; chaque espace indexé de la liste est appelé élément. Pour le composant DataGrid, chaque élément est composé de champs. Dans le code suivant, le contenu placé entre accolades (`{}`) est un élément :

  ```
  myDP = new Array({name:"Chris", price:"Priceless"}, {name:"Nigel",
    price:"Cheap"});
  ```

- Champ

  Identifiant indiquant le nom des colonnes dans les éléments. Correspond à la propriété `columnNames` dans la liste de colonnes. Habituellement, le composant List utilise les champs `label` et `data` ; dans le cas du composant DataGrid, il peut s'agir de n'importe quel identifiant. Dans le code suivant, les champs sont `name` et `price` :

  ```
  myDP = new Array({name:"Chris", price:"Priceless"}, {name:"Nigel",
    price:"Cheap"});
  ```

La vue comporte trois parties principales :

- Ligne

  Il s'agit d'un objet d'affichage chargé du rendu des éléments de la grille grâce à la disposition des cellules. Chaque ligne est disposée horizontalement sous la ligne précédente.

- Colonne

  Les colonnes sont les champs affichés dans une grille ; chaque champ correspond à la propriété `columnName` de chaque colonne.

  Chaque colonne est un objet d'affichage (occurrence de la classe DataGridColumn) chargé d'afficher chaque colonne (par exemple largeur, couleur, taille, etc.).

  Il existe trois méthodes pour ajouter des colonnes à une grille de données : l'affectation d'un objet DataProvider à `DataGrid.dataProvider` (génère automatiquement une colonne pour chaque champ dans le premier élément), la définition de la propriété `DataGrid.columnNames` pour désigner les champs à afficher, ou l'utilisation du constructeur de la classe DataGridColumn pour créer des colonnes, suivi de l'appel à la méthode `DataGrid.addColumn()` pour les ajouter à la grille.

  Pour formater les colonnes, vous pouvez définir les propriétés de style de toute la grille de données ou définir des objets DataGridColumn, configurer leur format de style individuellement, puis les ajouter à la grille de données.

- Cellule

  Il s'agit d'un objet d'affichage chargé du rendu de chaque champ de chaque élément. Pour communiquer avec la grille de données, ces composants doivent implémenter l'API CellRenderer (voir « API CellRenderer », à la page 111). Dans une grille de données de base, une cellule est un objet TextField ActionScript intégré.

Paramètres du composant DataGrid

Vous pouvez définir les paramètres de programmation suivants pour chaque occurrence du composant DataGrid dans l'inspecteur des propriétés ou des composants :

editable est une valeur booléenne indiquant si la grille est modifiable (`true`) ou non (`false`). La valeur par défaut est `false`.

multipleSelection est une valeur booléenne indiquant si plusieurs éléments peuvent être sélectionnés (`true`) ou non (`false`). La valeur par défaut est `false`.

rowHeight indique la hauteur de chaque ligne, en pixels. La modification de la taille de la police n'affecte pas la hauteur de la ligne. La valeur par défaut est 20.

Vous pouvez contrôler ces options et d'autres options du composant DataGrid à l'aide des propriétés, méthodes et événements d'ActionScript. Pour plus d'informations, voir « Classe DataGrid », à la page 275.

Création d'une application avec le composant DataGrid

Pour créer une application avec le composant DataGrid, commencez par déterminer d'où proviennent vos données. Elles peuvent provenir d'un jeu d'enregistrements alimenté par une requête de base de données dans Adobe ColdFusion, Java ou .Net à l'aide du système Flash Remoting. Elles peuvent également provenir d'un ensemble de données ou d'un tableau. Pour les intégrer à une grille, vous devez définir la propriété `DataGrid.dataProvider` (jeu d'enregistrements ou de données ou tableau correspondant). Vous pouvez également utiliser les méthodes des classes DataGrid et DataGridColumn pour créer les données localement. Un objet Array (tableau) se trouvant dans la même image qu'un composant DataGrid copie les méthodes, propriétés et événements de l'API DataProvider.

REMARQUE

Lorsque vous liez des données au composant DataGrid à l'aide des composants Data, l'objet lie les colonnes vers l'arrière (comme lorsque vous passez en boucle sur un objet ou un tableau). Par conséquent, pour ordonner différemment les données dans le composant DataGrid, vous devez définir des colonnes de façon explicite.

Pour ajouter un composant DataGrid à une application à l'aide du système Flash Remoting, procédez comme suit :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Dans le panneau Composants, double-cliquez sur le composant DataGrid pour l'ajouter sur la scène.
3. Dans l'inspecteur Propriétés, entrez le nom d'occurrence **myDataGrid**.
4. Entrez le code suivant dans le panneau Actions, sur l'image 1 :

   ```
   myDataGrid.dataProvider = recordSetInstance;
   ```

   Le jeu d'enregistrements Flash Remoting `recordSetInstance` est affecté à la propriété `dataProvider` de `myDataGrid`.
5. Sélectionnez Contrôle > Tester l'animation.

Pour ajouter un composant DataGrid à une application à l'aide d'un fournisseur de données local, procédez comme suit :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Dans le panneau Composants, double-cliquez sur le composant DataGrid pour l'ajouter sur la scène.
3. Dans l'inspecteur Propriétés, entrez le nom d'occurrence **myDataGrid**.
4. Entrez le code suivant dans le panneau Actions, sur l'image 1 :

   ```
   myDP = new Array({name:"Chris", price:"Priceless"}, {name:"Nigel",
     price:"Cheap"});
   myDataGrid.dataProvider = myDP;
   ```

   Les champs `name` et `price` sont utilisés comme en-têtes de colonne, et leurs valeurs remplissent chaque ligne des cellules.
5. Choisissez Contrôle> Tester l'animation.

Pour spécifier des colonnes et ajouter un tri pour un composant DataGrid dans une application :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Dans le panneau Composants, double-cliquez sur le composant DataGrid pour l'ajouter sur la scène.
3. Dans l'inspecteur Propriétés, entrez le nom d'occurrence **myDataGrid**.

4. Entrez le code suivant dans le panneau Actions, sur l'image 1 :

```
var myDataGrid:mx.controls.DataGrid;

// Création de colonnes pour activer le tri des données.
myDataGrid.addColumn("name");
myDataGrid.addColumn("score");

var myDP_array:Array = new Array({name:"Clark", score:3135},
  {name:"Bruce", score:403}, {name:"Peter", score:25})

myDataGrid.dataProvider = myDP_array;

// Création d'un objet écouteur pour DataGrid.
var listener_obj:Object = new Object();
listener_obj.headerRelease = function(evt_obj:Object) {
 switch (evt_obj.target.columns[evt_obj.columnIndex].columnName) {
 case "name" :
  myDP_array.sortOn("name", Array.CASEINSENSITIVE);
  break;
 case "score" :
  myDP_array.sortOn("score", Array.NUMERIC);
  break;
 }
};

// Ajout de l'écouteur au DataGrid.
myDataGrid.addEventListener("headerRelease", listener_obj);
```

5. Choisissez Contrôle> Tester l'animation.

Pour créer une occurrence du composant DataGrid à l'aide d'ActionScript :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser le composant DataGrid du panneau Composants jusqu'à la bibliothèque du document actuel.

   Cette opération permet d'ajouter le composant à la bibliothèque, mais elle ne permet pas de le rendre visible dans l'application.
3. Sélectionnez la première image dans le scénario principal, ouvrez le panneau Actions et saisissez le code suivant :

```
this.createClassObject(mx.controls.DataGrid, "my_dg", 10,
  {columnNames:["name", "score"]});
my_dg.setSize(140, 100);
my_dg.move(10, 40);
```

   Ce script utilise la méthode `UIObject.createClassObject()` pour créer l'occurrence du composant DataGrid, puis dimensionne et positionne la grille.

4. Créez maintenant un tableau, ajoutez-y des données et identifiez-le comme fournisseur de données de la grille :

```
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});
my_dg.dataProvider = myDP_array;
```

5. Choisissez Contrôle> Tester l'animation.

Stratégies de performance DataGrid

Les performances peuvent rapidement poser un problème important si vous utilisez le composant DataGrid, car la taille des données affichées peut être évolutive. Un composant DataGrid affichant une centaine de lignes sur un ordinateur assez performant doté d'une connexion rapide à la source de données peut vous sembler très utile. Un mois plus tard, en revanche, lorsque les données ont atteint plusieurs milliers de lignes, l'expérience est bien différente. L'utilisateur peut également posséder un ordinateur moins performant, doté d'une connexion lente à la source de données.

Voici quelques suggestions pour éviter des problèmes de performance courants lors de l'utilisation du composant DataGrid.

- Créer et lier une structure de données au lieu d'ajouter directement des colonnes.

  Deux méthodes permettent d'ajouter des colonnes et des données au composant DataGrid : en liant une structure de données prête à l'emploi (tableau d'objets) via la propriété `DataGrid.dataProvider` ou en utilisant les méthodes de la classe DataGrid telles que `DataGrid.addColumn()` et `DataGrid.addItem()`. Chaque fois que cela est possible, il est conseillé de lier une structure de données prête à l'emploi à l'aide de la propriété `DataGrid.dataProvider`, car elle permet au composant DataGrid de créer toutes les colonnes nécessaires avant de les tracer à l'écran.

  Vous pouvez être tenté de créer une boucle `for` pour appeler la méthode `DataGrid.addColumn()` pour toutes les colonnes nécessaires. Même si cette méthode semble simple et évidente, ne l'utilisez-pas. Chaque fois que la méthode `DataGrid.addColumn` est appelée, le composant DataGrid ajoute des écouteurs d'événements, trie et se redessine pour présenter la nouvelle colonne. Lors de la création de 20 colonnes à l'aide de `DataGrid.addColumn`, le composant DataGrid se trie et se redessine 20 fois inutilement. La création de votre structure de données dans ActionScript ne requiert aucun rendu ni événement. Si vous l'affectez à la propriété `dataProvider` du composant DataGrid, l'ensemble du dessin est terminé du même coup.

- Fournir un mécanisme de développement pour les données détaillées.

  L'interface du composant DataGrid permet à l'utilisateur d'effectuer des recherches rapides et donc de rechercher des données plus détaillées. Fournissez seulement les données nécessaires à la recherche initiale, puis dans une seconde étape, des informations détaillées correspondant à une cellule ou à une ligne particulière. Cette procédure minimise non seulement les données initiales requises pour remplir le composant DataGrid, mais également la quantité d'informations que l'utilisateur doit lire pour trouver ce qu'il recherche. Une fois qu'une ligne ou un élément intéressant est sélectionné dans la grille de données, un deuxième appel à la source de données peut être effectué pour obtenir les détails afférents. L'affichage de ces détails peut être de meilleure qualité dans un autre mécanisme d'interface utilisateur (collection de champs de texte à plusieurs lignes et graphiques, par exemple).
- Eviter les cycles de manipulation de données entre la source de données et le composant DataGrid.

  Si cela est possible et répond aux besoins en matière de base de données à long terme, le stockage des données dans le même format et le même ordre que leur affichage peut économiser du temps de traitement et d'allocation mémoire inutile sur l'ordinateur de l'utilisateur et accélérer le temps de réponse du composant DataGrid.
- Eviter les requêtes qui renvoient chaque ligne dans la base de données.

  Les utilisateurs souhaitent rarement consulter tous les enregistrements disponibles à chaque fois qu'ils accèdent à leurs données. Il est important de bien comprendre ce que recherchent les utilisateurs de vos données et de leur donner la possibilité d'affiner leurs recherches. S'ils consultent généralement uniquement les enregistrements les plus récents d'une semaine en question pour un sujet donné, affichez ce groupe plus petit de données comme groupe par défaut avec la possibilité d'élargir l'affichage des données.

  Pensez éventuellement à mettre en pages de nombreuses données afin de limiter leur taille en fournissant un sous-ensemble de données qui pourrait être renvoyé normalement suite à une requête. Par exemple, au lieu d'afficher l'intégralité des 10 000 lignes de données pouvant être renvoyées par votre base de données, un sous-ensemble des 20 premières lignes peut être appelé, et des boutons de navigation supplémentaires peuvent déclencher un appel remplissant le composant DataGrid avec les 20 enregistrements suivants.

- Séparer le traitement des données du traitement CellRenderer.

  Les API CellRenderer permettent d'afficher des cellules au contenu personnalisé dans un composant DataGrid. Pour des raisons de fonctionnement, vous pouvez être tenu de remplir le composant DataGrid avec un composant ComboBox ou une autre commande de l'interface utilisateur de façon conditionnelle. Par exemple, en fonction d'une sélection dans la colonne deux, vous devez de nouveau remplir ou sélectionner automatiquement des options dans la colonne quatre. Dans la mesure du possible, il est important de séparer cette logique conditionnelle et le nouveau remplissage des commandes du processus de rendu du contenu de la cellule. Chaque fois que la souris survole la cellule, cette dernière est sélectionnée ou les données sont modifiées. Il est probable que le contenu de la cellule ou la cellule tout entière soit redessinée et mise à jour. En d'autres termes, le code que vous insérez dans CellRenderer est exécuté sans cesse. Vous devez donc alléger le traitement dans CellRenderer dans la mesure du possible. Si vous devez ajouter du code à CellRenderer, il est préférable d'appeler une fonction à partir de CellRenderer qui soit capable de détecter les mises à jour nécessaires et de les effectuer de la façon la plus efficace possible.
- Utiliser `UIObject.doLater()` pour accéder aux propriétés dès que le composant DataGrid est dessiné.

  Une instance du composant DataGrid doit être dessinée et avoir chargé les données pour que vous puissiez accéder à toutes ses propriétés (notamment `focusedCell`, etc.). Etant donné qu'il n'existe pas d'événement « complete » pour un composant DataGrid, vous pouvez utiliser à la place `UIObject.doLater()` pour appeler une fonction qui accède aux propriétés de ce composant. `UIObject.doLater()` exécute cette fonction dès que les propriétés du composant DataGrid sont disponibles. Pour consulter un exemple, reportez-vous à `DataGrid.focusedCell`.

Personnalisation du composant DataGrid

Vous pouvez transformer un composant DataGrid horizontalement et verticalement durant la programmation et à l'exécution. Lors de la programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez la méthode `setSize()` (voir `UIObject.setSize()`). Lorsque aucune barre de défilement horizontale n'est présente, la largeur des colonnes s'ajuste proportionnellement. Si une modification de la taille des colonnes (et donc des cellules) intervient, le texte des cellules peut être tronqué.

Utilisation des styles avec le composant DataGrid

Vous pouvez définir des propriétés de style afin de modifier l'aspect d'un composant DataGrid. Le composant DataGrid hérite des styles du composant List. (Voir la section « Utilisation des styles avec le composant List », à la page 797). Le composant DataGrid prend également en charge les styles suivants :

Style	Thème	Description
`backgroundColor`	Les deux	Couleur d'arrière-plan, qui peut être définie pour l'ensemble de la grille ou pour chaque colonne.
`backgroundDisabledColor`	Les deux	Couleur d'arrière-plan lorsque la propriété `enabled` du composant est définie sur `"false"`. La valeur par défaut est 0xDDDDDD (gris moyen).
`borderStyle`	Les deux	Le composant DataGrid utilise une occurrence RectBorder en tant que bordure et répond au styles définis sur cette classe. Voir « Classe RectBorder », à la page 1111. La valeur par défaut du style de bordure est `"inset"`.
`headerColor`	Les deux	Couleur des en-têtes de colonnes. La valeur par défaut est 0xEAEAEA (gris clair).
`headerStyle`	Les deux	Déclaration de style CSS pour l'en-tête de colonne, pouvant être appliquée à une grille ou à une colonne pour personnaliser les styles d'en-tête.
`color`	Les deux	Couleur du texte. La valeur par défaut est 0x0B333C pour le thème Halo et vide pour le thème Sample.
`disabledColor`	Les deux	Couleur du texte lorsque le composant est désactivé. La couleur par défaut est 0x848384 (gris foncé).

`embedFonts`	Les deux	Valeur booléenne qui indique si la police spécifiée dans `fontFamily` est de type intégré. Ce style doit être défini sur `true` si `fontFamily` fait référence à une police intégrée. Par exemple (à l'aide d'une occurrence du composant DataGrid `my_dg`) :
		`my_dg.setStyle("fontFamily", "yourFont");`
		`my_dg.embedFonts=true;`
		Dans le cas contraire, la police intégrée n'est pas utilisée. Si ce style est défini sur `true` et que `fontFamily` ne fait pas référence à une police intégrée, aucun texte n'est affiché. La valeur par défaut est `false`.

`fontFamily`	Les deux	Nom de la police du texte. La valeur par défaut est `"_sans"`.

Style	Thème	Description
`fontSize`	Les deux	Taille de la police, en points. La valeur par défaut est 10.
`fontStyle`	Les deux	Style de police : `"normal"` ou `"italic"`. La valeur par défaut est `"normal"`.
`fontWeight`	Les deux	Epaisseur de la police : `"none"` ou `"bold"`. La valeur par défaut est `"none"`. Tous les composants peuvent également accepter la valeur « `normal` » au lieu de « `none` » pendant un appel de `setStyle()`, mais les prochains appels de `getStyle()` renvoient « `none` ».
`textAlign`	Les deux	Alignement du texte : « `left` », « `right` » ou « `center` ». La valeur par défaut est « `left` ».
`textDecoration`	Les deux	Décoration du texte : `"none"` ou `"underline"`. La valeur par défaut est `"none"`.
`vGridLines`	Les deux	Valeur booléenne indiquant si les lignes verticales de la grille doivent être affichées (`true`) ou non (`false`). La valeur par défaut est `true`.
`hGridLines`	Les deux	Valeur booléenne indiquant si les lignes horizontales de la grille doivent être affichées (`true`) ou non (`false`). La valeur par défaut est `false`.
`vGridLineColor`	Les deux	Couleur des lignes verticales de la grille. La valeur par défaut est 0x666666 (gris moyen).
`hGridLineColor`	Les deux	Couleur des lignes horizontales de la grille. La valeur par défaut est 0x666666 (gris moyen).

Définition des styles pour une colonne individuelle

Des styles de couleur et de texte peuvent être définis pour l'ensemble de la grille ou pour une colonne. La syntaxe suivante permet de définir un style pour une colonne particulière :

```
grid.getColumnAt(3).setStyle("backgroundColor", 0xFF00AA);
```

Définition de styles d'en-tête

Vous pouvez définir des styles d'en-tête via `headerStyle`, elle-même propriété de style. Pour ce faire, créez une occurrence de `CSSStyleDeclaration`, définissez les propriétés appropriées sur cette occurrence pour l'en-tête, puis affectez `CSSStyleDeclaration` à la propriété `headerStyle`, comme dans l'exemple suivant.

```
import mx.styles.CSSStyleDeclaration;
var headerStyles = new CSSStyleDeclaration();
headerStyles.setStyle("fontStyle", "italic");
grid.setStyle("headerStyle", headerStyles);
```

Définition de styles pour tous les composants DataGrid d'un document

La classe DataGrid hérite de la classe List, qui elle-même hérite de la classe ScrollSelectList. Les propriétés de style de niveau classe sont définies sur la classe ScrollSelectList, étendue par le composant Menu et tous les composants de type List. Vous pouvez définir de nouvelles valeurs de style par défaut directement sur cette classe, ces nouveaux paramètres sont alors reflétés dans tous les composants concernés.

```
_global.styles.ScrollSelectList.setStyle("backgroundColor", 0xFF00AA);
```

Pour ne définir une propriété de style que sur les composants DataGrid, vous pouvez créer une occurrence de `CSSStyleDeclaration` et la stocker dans `_global.styles.DataGrid`.

```
import mx.styles.CSSStyleDeclaration;
if (_global.styles.DataGrid == undefined) {
  _global.styles.DataGrid = new CSSStyleDeclaration();
}
_global.styles.DataGrid.setStyle("backgroundColor", 0xFF00AA);
```

Lors de la création d'une déclaration de style de niveau classe, toutes les valeurs par défaut fournies par la déclaration `ScrollSelectList` sont perdues, y compris `backgroundColor`, requise pour la prise en charge des événements liés à la souris. Pour créer une déclaration de style de niveau classe et conserver les valeurs par défaut, utilisez une boucle `for..in` pour copier les anciens paramètres dans la nouvelle déclaration.

```
var source = _global.styles.ScrollSelectList;
var target = _global.styles.DataGrid;
for (var style in source) {
  target.setStyle(style, source.getStyle(style));
}
```

Pour plus d'informations sur les styles de niveau classe, reportez-vous à « Définition de styles pour une classe de composants » dans *Utilisation des composants ActionScript 2.0*.

Utilisation d'enveloppes avec le composant DataGrid

Les enveloppes utilisées par le composant DataGrid pour représenter ses états virtuels sont incluses dans les sous-composants qui forment la grille de données (barres de défilement et RectBorder). Pour plus d'informations sur les enveloppes, reportez-vous à « Utilisation d'enveloppes avec le composant UIScrollBar », à la page 1448 et à « Classe RectBorder », à la page 1111.

Classe DataGrid

Héritage MovieClip > Classe UIObject > Classe UIComponent > View > ScrollView > ScrollSelectList > Composant List > DataGrid

Nom de classe ActionScript mx.controls.DataGrid

Toutes les classes de composants ont une propriété `version` qui correspond à une propriété de classe. Les propriétés de classe ne sont disponibles que pour la classe elle-même. La propriété `version` renvoie une chaîne qui indique la version du composant. Pour accéder à cette propriété, utilisez le code suivant :

```
trace(mx.controls.DataGrid.version);
```

REMARQUE

Le code `trace(myDataGridInstance.version);` renvoie `undefined`.

Récapitulatif des méthodes de la classe DataGrid

Le tableau suivant présente les méthodes de la classe DataGrid.

Méthode	Description
`DataGrid.addColumn()`	Ajoute une colonne à la grille de données.
`DataGrid.addColumnAt()`	Ajoute une colonne à la grille de données à l'emplacement spécifié.
`DataGrid.addItem()`	Ajoute un élément à la grille de données.
`DataGrid.addItemAt()`	Ajoute un élément à la grille de données à l'emplacement spécifié.
`DataGrid.editField()`	Remplace les données d'une cellule à un emplacement spécifié.

Méthode	Description
DataGrid.getColumnAt()	Obtient une référence à une colonne à l'emplacement spécifié.
DataGrid.getColumnIndex()	Obtient une référence à l'objet DataGridColumn au niveau de l'index spécifié.
DataGrid.removeAllColumns()	Supprime toutes les colonnes d'une grille de données.
DataGrid.removeColumnAt()	Supprime une colonne d'une grille de données à l'emplacement spécifié.
DataGrid.replaceItemAt()	Remplace un élément par un autre à l'emplacement spécifié.
DataGrid.spaceColumnsEqually()	Espace les colonnes de manière égale.

Méthodes héritées de la classe UIObject

Le tableau suivant énumère les méthodes de la classe DataGrid héritées de la classe UIObject. Pour appeler ces méthodes, utilisez le formulaire *dataGridInstance.methodName*.

Méthode	Description
UIObject.createClassObject()	Crée un objet dans la classe spécifiée.
UIObject.createObject()	Crée un sous-objet dans un objet.
UIObject.destroyObject()	Détruit une occurrence de composant.
UIObject.doLater()	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
UIObject.getStyle()	Obtient la propriété de style de l'objet ou de la déclaration de style.
UIObject.invalidate()	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
UIObject.move()	Déplace l'objet à l'emplacement demandé.
UIObject.redraw()	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
UIObject.setSize()	Redimensionne l'objet à la taille demandée.
UIObject.setSkin()	Définit une enveloppe dans l'objet.
UIObject.setStyle()	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant énumère les méthodes de la classe DataGrid héritées de la classe UIComponent. Pour appeler ces méthodes, utilisez le formulaire *dataGridInstance.methodName*.

Méthode	Description
UIComponent.getFocus()	Renvoie une référence à l'objet ayant le focus.
UIComponent.setFocus()	Attribue le focus à l'occurrence de composant.

Méthodes héritées de la classe List

Le tableau suivant énumère les méthodes de la classe DataGrid héritées de la classe List. Pour appeler ces méthodes, utilisez le formulaire *dataGridInstance.methodName*.

Méthode	Description
List.addItem()	Ajoute un élément à la fin de la liste.
List.addItemAt()	Ajoute un élément à la liste, à l'index spécifié.
List.getItemAt()	Renvoie l'élément à l'emplacement d'index spécifié.
List.removeAll()	Supprime tous les éléments de la liste.
List.removeItemAt()	Supprime l'élément à l'index spécifié.
List.replaceItemAt()	Remplace l'élément par un autre, à l'index spécifié.
List.setPropertiesAt()	Applique les propriétés spécifiées à l'élément donné.
List.sortItems()	Trie les éléments de la liste à l'aide de la fonction de comparaison spécifiée.
List.sortItemsBy()	Trie les éléments de la liste à l'aide d'une propriété donnée.

Récapitulatif des propriétés de la classe DataGrid

Le tableau suivant présente les propriétés de la classe DataGrid.

Propriété	Description
DataGrid.columnCount	Lecture seule ; nombre de colonnes affichées.
DataGrid.columnNames	Tableau des noms de champs (affichés sous forme de colonnes) de chaque élément.
DataGrid.dataProvider	Modèle de données d'une grille de données.

Propriété	Description
`DataGrid.editable`	Valeur booléenne indiquant si la grille de données est modifiable (`true`) ou non (`false`).
`DataGrid.focusedCell`	Définit la cellule qui a le focus.
`DataGrid.headerHeight`	Hauteur des en-têtes de colonnes, en pixels.
`DataGrid.hScrollPolicy`	Indique si une barre de défilement horizontale est présente (`"on"`), absente (`"off"`) ou apparaît lorsque cela est nécessaire (`"auto"`).
`DataGrid.resizableColumns`	Valeur booléenne indiquant si les colonnes peuvent être redimensionnées (`true`) ou non (`false`).
`DataGrid.selectable`	Valeur booléenne indiquant si la grille de données peut être sélectionnée (`true`) ou non (`false`).
`DataGrid.showHeaders`	Valeur booléenne indiquant si les en-têtes de colonnes sont visibles (`true`) ou non (`false`).
`DataGrid.sortableColumns`	Valeur booléenne indiquant si les colonnes peuvent être triées (`true`) ou non (`false`).

Propriétés héritées de la classe UIObject

Le tableau suivant énumère les propriétés de la classe DataGrid héritées de la classe UIObject. Lors de l'accès à ces propriétés à partir de l'objet DataGrid, utilisez le formulaire *`dataGridInstance.propertyName`*.

Propriété	Description
`UIObject.bottom`	Position du bord inférieur de l'objet par rapport au bord inférieur de son parent. Lecture seule.
`UIObject.height`	Hauteur de l'objet, en pixels. Lecture seule.
`UIObject.left`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.right`	Position du bord droit de l'objet par rapport au bord droit de son parent. Lecture seule.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Position du bord supérieur de l'objet par rapport à son parent. Lecture seule.

Propriété	Description
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Largeur de l'objet, en pixels. Lecture seule.
`UIObject.x`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.y`	Bord supérieur de l'objet, en pixels. Lecture seule.

Propriétés héritées de la classe UIComponent

Le tableau suivant énumère les propriétés de la classe DataGrid héritées de la classe UIComponent. Lors de l'accès à ces propriétés à partir de l'objet DataGrid, utilisez le formulaire *`dataGridInstance.propertyName`*.

Propriété	Description
`UIComponent.enabled`	Indique si le composant peut recevoir le focus et la saisie.
`UIComponent.tabIndex`	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Propriétés héritées de la classe List

Le tableau suivant énumère les propriétés de la classe DataGrid héritées de la classe List. Lors de l'accès à ces propriétés à partir de l'objet DataGrid, utilisez le formulaire *`dataGridInstance.propertyName`*.

Propriété	Description
`List.cellRenderer`	Affecte la classe ou le symbole à utiliser pour afficher chaque ligne de la liste.
`List.dataProvider`	Source des éléments de la liste.
`List.hPosition`	Position horizontale de la liste.
`List.hScrollPolicy`	Indique si la barre de défilement horizontale est affichée (`"on"`) ou non (`"off"`).
`List.iconField`	Champ situé à l'intérieur de chaque élément et utilisé pour spécifier les icônes.
`List.iconFunction`	Fonction qui détermine les icônes à utiliser.
`List.labelField`	Spécifie un champ dans chaque élément, à utiliser comme texte d'étiquette.
`List.labelFunction`	Fonction qui détermine les champs de chaque élément à utiliser pour le texte d'étiquette.

Propriété	Description
`List.length`	Nombre d'éléments de la liste. Cette propriété est en lecture seule.
`List.maxHPosition`	Nombre de pixels que la liste peut faire défiler à droite, lorsque `List.hScrollPolicy` est défini sur `"on"`.
`List.multipleSelection`	Indique si la sélection multiple est autorisée dans la liste (`true`) ou non (`false`).
`List.rowCount`	Nombre de lignes au moins partiellement visibles dans la liste.
`List.rowHeight`	Hauteur de chaque ligne de la liste, en pixels.
`List.selectable`	Indique si la liste peut être sélectionnée (`true`) ou non (`false`).
`List.selectedIndex`	Index d'une sélection dans une liste à sélection unique.
`List.selectedIndices`	Tableau des éléments sélectionnés dans une liste à sélection multiple.
`List.selectedItem`	Elément sélectionné dans une liste à sélection unique. Cette propriété est en lecture seule.
`List.selectedItems`	Objets sélectionnés dans une liste à sélection multiple. Cette propriété est en lecture seule.
`List.vPosition`	Fait défiler la liste pour que le premier élément visible soit le numéro affecté.
`List.vScrollPolicy`	Indique si la barre de défilement verticale est affichée (`"on"`), ne l'est pas (`"off"`) ou est affichée si nécessaire (`"auto"`).

Récapitulatif des événements de la classe DataGrid

Le tableau suivant présente les événements de la classe DataGrid.

Evénement	Description
`DataGrid.cellEdit`	Diffusé lorsque la valeur de la cellule a changé.
`DataGrid.cellFocusIn`	Diffusé lorsqu'une cellule reçoit le focus.
`DataGrid.cellFocusOut`	Diffusé lorsqu'une cellule perd le focus.
`DataGrid.cellPress`	Diffusé lorsque l'utilisateur clique sur une cellule.
`DataGrid.change`	Diffusé lorsqu'un élément a été sélectionné.
`DataGrid.columnStretch`	Diffusé lorsqu'un utilisateur redimensionne une colonne horizontalement.
`DataGrid.headerRelease`	Diffusé lorsqu'un utilisateur clique et relâche le bouton de la souris sur un en-tête.

Evénements hérités de la classe UIObject

Le tableau suivant énumère les événements de la classe DataGrid hérités de la classe UIObject.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant énumère les événements de la classe DataGrid hérités de la classe UIComponent.

Evénement	Description
`UIComponent.focusIn`	Diffusé lorsqu'un objet reçoit le focus.
`UIComponent.focusOut`	Diffusé lorsqu'un objet perd le focus.
`UIComponent.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIComponent.keyUp`	Diffusé lorsqu'une touche est relâchée.

Evénements hérités de la classe List

Le tableau suivant énumère les événements de la classe DataGrid hérités de la classe List.

Evénement	Description
`List.change`	Diffusé lorsqu'une interaction de l'utilisateur entraîne une modification de la sélection.
`List.itemRollOut`	Diffusé lorsque le pointeur de la souris survole, puis quitte un élément de la liste.

Evénement	Description
`List.itemRollOver`	Diffusé lorsque le pointeur de la souris survole des éléments de la liste.
`List.scroll`	Diffusé lorsqu'une liste défile.

DataGrid.addColumn()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.addColumn(*dataGridColumn*)

myDataGrid.addColumn(*name*)

Paramètres

dataGridColumn Occurrence de la classe DataGridColumn.

name Chaîne indiquant le nom d'un nouvel objet DataGridColumn à insérer.

Valeur renvoyée

Référence à l'objet DataGridColumn qui a été ajouté ou chaîne indiquant le nom de la nouvelle colonne.

Description

Méthode qui ajoute une nouvelle colonne à la fin de la grille de données. Pour plus d'informations, voir « Classe DataGridColumn », à la page 315.

Exemple

Cet exemple illustre trois façons différentes de créer des colonnes pour un composant DataGrid. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` se trouve sur la scène, collez le code suivant dans la première image du scénario principal (vous remarquerez que la classe DataGridColumn est d'abord importée) :

```
import mx.controls.gridclasses.DataGridColumn;

var my_dg:mx.controls.DataGrid;
my_dg.setSize(320, 240);
```

```
// Ajout de colonnes à la grille.
my_dg.addColumn("Red");

// Ajout d'une autre colonne à la grille.
my_dg.addColumn(new DataGridColumn("Green"));

// Ajout d'une troisième colonne à la grille.
var blue_dgc:DataGridColumn = new DataGridColumn("Blue");
blue_dgc.width = 140;
blue_dgc.headerText = "Blue Column:";
my_dg.addColumn(blue_dgc);
```

DataGrid.addColumnAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.addColumnAt(*index*, *name*)

myDataGrid.addColumnAt(*index*, *dataGridColumn*)

Paramètres

index Position d'index à laquelle l'objet DataGridColumn est ajouté. La première position est 0.

name Chaîne indiquant le nom de l'objet DataGridColumn.

dataGridColumn Occurrence de la classe DataGridColumn.

Valeur renvoyée

Référence à l'objet DataGridColumn qui a été ajouté ou chaîne indiquant le nom de la nouvelle colonne.

Description

Méthode qui ajoute une nouvelle colonne à l'emplacement spécifié. Les colonnes sont déplacées vers la droite et leur index est incrémenté. Pour plus d'informations, voir « Classe DataGridColumn », à la page 315.

Exemple

Cet exemple illustre deux façons d'utiliser `addColumnAt()` et définit les largeurs de colonnes. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` se trouve sur la scène, collez le code suivant dans la première image du scénario principal (vous remarquerez que la classe DataGridColumn est d'abord importée) :

```
import mx.controls.gridclasses.DataGridColumn;

var my_dg:mx.controls.DataGrid;
my_dg.setSize(320, 240);

// Ajout de colonnes à la grille.
my_dg.addColumnAt(0, "Orange");
var orange_dgc:DataGridColumn = my_dg.getColumnAt(0);
orange_dgc.width = 125;

var blue_dgc:DataGridColumn = new DataGridColumn("Blue");
blue_dgc.width = 75;
my_dg.addColumnAt(1, blue_dgc);
```

DataGrid.addItem()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.addItem(*item*)

Paramètres

item Occurrence d'un objet à ajouter à la grille.

Valeur renvoyée

Une référence à l'occurrence qui a été ajoutée.

Description

Méthode qui ajoute un élément à la fin de la grille (après le dernier index d'élément).

REMARQUE	Cette méthode diffère de la méthode `List.addItem()` dans la mesure où c'est un objet, et non une chaîne, qui est transmis.

Exemple

Cet exemple crée une colonne avec l'en-tête « name », puis insère la valeur `item_obj` pour « name ». Vous remarquerez que la valeur « age » est ignorée car seule la colonne « name » a été définie. Si vous ne spécifiez pas de colonne (supprimez la ligne `addColumn`), le composant DataGrid crée automatiquement les colonnes appropriées. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` se trouve sur la scène, collez le code suivant dans la première image du scénario principal :

```
// Ajout de colonnes à la grille et ajout de données.
my_dg.addColumn("name");

var item_obj:Object = {name:"Jim", age:30};
my_dg.addItem(item_obj);
```

DataGrid.addItemAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.addItemAt(*index*, *item*)

Paramètres

index Emplacement d'index (parmi les nœuds enfants) auquel le nœud doit être ajouté. La première position est 0.

item Chaîne qui affiche le nœud.

Valeur renvoyée

Une référence à l'occurrence d'objet qui a été ajoutée.

Description

Méthode qui ajoute un élément à la grille à l'emplacement spécifié.

Exemple

Cet exemple crée une colonne avec l'en-tête « name », remplit la colonne depuis un tableau et ajoute ensuite le nom « Chase » dans la première ligne. Vous remarquerez que la valeur « age » est ignorée car seule la colonne « name » a été définie. Si vous ne spécifiez pas de colonne (supprimez la ligne `addColumn`), le composant DataGrid crée automatiquement les colonnes appropriées. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` se trouve sur la scène, collez le code suivant dans la première image du scénario principal :

```
var my_dg:mx.controls.DataGrid;

// Ajout de colonnes à la grille et ajout de données.
my_dg.addColumn("name");

var myDP_array:Array = new Array({name:"John", age:33}, {name:"Jose",
  age:41});
my_dg.dataProvider = myDP_array;

var item_obj:Object = {name:"Chase", age:30};
my_dg.addItemAt(0, item_obj);
```

DataGrid.cellEdit

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
listenerObject = new Object();
listenerObject.cellEdit = function(eventObject){
  // Insertion du code ici.
}
myDataGridInstance.addEventListener("cellEdit", listenerObject)
```

Description

Evénement ; diffusé à tous les objets écouteurs enregistrés lorsque la valeur d'une cellule change.

Les composants utilisent un modèle dispatcher(diffuseur)/écouteur d'événement. Le composant DataGrid distribue un événement `cellEdit` lorsque la valeur d'une cellule a été modifiée ; l'événement est géré par une fonction (également appelée *gestionnaire*) associée à un objet écouteur (*`listenerObject`*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`eventObject`*) au gestionnaire. Chaque objet événement est doté de propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. L'objet événement de l'événement `DataGrid.cellEdit` possède quatre propriétés supplémentaires :

`columnIndex` Nombre indiquant l'index de la colonne cible.

`itemIndex` Nombre indiquant l'index de la ligne cible.

`oldValue` Valeur précédente de la cellule.

`type` Chaîne `"cellEdit"`.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `myDataGridListener` est défini, puis transmis à la méthode `myDataGrid.addEventListener()` en tant que second paramètre. L'objet événement est capturé par le gestionnaire `cellEdit` dans le paramètre *`eventObject`*. Lorsque l'événement `cellEdit` est diffusé (une fois que vous avez modifié une valeur « score » et appuyé sur Entrée), une instruction `trace` est envoyée au panneau Sortie. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` se trouve sur la scène, collez le code suivant dans la première image du scénario principal :

```
my_dg.setSize(320, 240);
my_dg.editable = true;

// Ajout de colonnes et transformation de la première en une colonne
// non modifiable.
my_dg.addColumn("name");
my_dg.getColumnAt(0).editable = false;
my_dg.addColumn("score");

var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});
```

```
// Définition de la source de données du composant DataGrid.
my_dg.dataProvider = myDP_array;

// Création d'un objet écouteur.
var myListener_obj:Object = new Object();
myListener_obj.cellEdit = function(evt_obj:Object) {
 // Récupération de l'emplacement de la cellule modifiée.
 var cell_obj:Object = "("+evt_obj.columnIndex+", "+evt_obj.itemIndex+")";
 // Récupération de la valeur de la cellule ayant été modifiée.
 var value_obj:Object = evt_obj.target.selectedItem.score;
 trace("The value of the cell at "+cell_obj+" has changed to "+value_obj);
};

// Ajout de l'objet écouteur.
my_dg.addEventListener("cellEdit", myListener_obj);
```

DataGrid.cellFocusIn

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
listenerObject = new Object();
listenerObject.cellFocusIn = function(eventObject){
  // Insertion du code ici.
}
myDataGridInstance.addEventListener("cellFocusIn", listenerObject)
```

Description

Evénement diffusé à tous les objets écouteurs enregistrés lorsqu'une cellule reçoit le focus. Cet événement est diffusé une fois que les événements `editCell` et `cellFocusOut` d'une cellule préalablement modifiée sont diffusés.

Les composants utilisent un modèle dispatcher(diffuseur)/écouteur d'événement. Lorsqu'un composant DataGrid distribue un événement `cellFocusIn` qui est géré par une fonction (également appelée *gestionnaire*) associée à un objet écouteur (*`listenerObject`*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. L'objet événement de l'événement `DataGrid.cellFocusIn` possède trois propriétés supplémentaires :

`columnIndex` Nombre indiquant l'index de la colonne cible.

`itemIndex` Nombre indiquant l'index de la ligne cible.

`type` Chaîne `"cellFocusIn"`.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `dgListener` est défini, puis transmis à la méthode `my_dg.addEventListener()` en tant que second paramètre. Lorsque l'événement `cellFocusIn` est diffusé, une instruction `trace` est envoyée au panneau Sortie. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` se trouve sur la scène, collez le code suivant dans la première image du scénario principal :

```
// Configuration des exemples de données.
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});

my_dg.dataProvider = myDP_array;

// Transformation du composant DataGrid en composant modifiable.
my_dg.editable = true;

// Création d'un objet écouteur.
var dgListener:Object = new Object();
dgListener.cellFocusIn = function(evt_obj:Object) {
    var cell_str:String = "(" + evt_obj.columnIndex + ", " +
  evt_obj.itemIndex + ")";
    trace("The cell at " + cell_str + " has gained focus");
};

// Ajout de l'écouteur.
my_dg.addEventListener("cellFocusIn", dgListener);
```

REMARQUE

La grille doit être modifiable pour que le code mentionné ci-dessus fonctionne, et l'événement est diffusé uniquement pour les cellules modifiables. Par conséquent, si vous avez deux colonnes dont une seule est modifiable (« score », par exemple) et que vous avez cliqué sur une ligne dans la colonne « name », cet événement n'est pas déclenché.

DataGrid.cellFocusOut

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
listenerObject = new Object();
listenerObject.cellFocusOut = function(eventObject){
  // Insertion du code ici.
}
myDataGridInstance.addEventListener("cellFocusOut", listenerObject)
```

Description

Evénement diffusé à tous les objets écouteurs enregistrés chaque fois qu'un utilisateur quitte une cellule qui a le focus. Vous pouvez utiliser les propriétés de l'objet événement pour isoler la cellule qui a été quittée. Cet événement est diffusé après la diffusion de l'événement `cellEdit` et avant celle des événements `cellFocusIn` consécutifs par la cellule suivante.

Les composants utilisent un modèle d'événement dispatcher(diffuseur)/écouteur. Lorsqu'un composant DataGrid distribue un événement `cellFocusOut`, celui-ci est géré par une fonction (également appelée *gestionnaire*) associée à un objet écouteur que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`eventObject`*) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. L'objet événement de l'événement `DataGrid.cellFocusOut` possède trois propriétés supplémentaires :

`columnIndex` Nombre indiquant l'index de la colonne cible. La première position est 0.

`itemIndex` Nombre indiquant l'index de la ligne cible. La première position est 0.

`type` Chaîne `"cellFocusOut"`.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `dgListener` est défini, puis transmis à la méthode `my_dg.addEventListener()` en tant que second paramètre. Lorsque l'événement `cellFocusOut` est diffusé, une instruction `trace` est envoyée au panneau Sortie. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` se trouve sur la scène, collez le code suivant dans la première image du scénario principal :

```
// Configuration des exemples de données.
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});

my_dg.dataProvider = myDP_array;

// Transformation du composant DataGrid en composant modifiable.
my_dg.editable = true;

// Création d'un objet écouteur.
var dgListener:Object = new Object();
dgListener.cellFocusOut = function(evt_obj:Object) {
    var cell_str:String = "(" + evt_obj.columnIndex + ", " +
  evt_obj.itemIndex + ")";
    trace("The cell at " + cell_str + " has lost focus");
};

// Ajout de l'écouteur.
my_dg.addEventListener("cellFocusOut", dgListener);
```

REMARQUE

La grille doit être modifiable pour que le code mentionné ci-dessus fonctionne, et l'événement est diffusé uniquement pour les cellules modifiables. Par conséquent, si vous avez deux colonnes dont une seule est modifiable (« score », par exemple) et que vous avez cliqué sur une ligne dans la colonne « name », cet événement n'est pas déclenché.

DataGrid.cellPress

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
listenerObject = new Object();
listenerObject.cellPress = function(eventObject){
  // Insertion du code ici.
}
myDataGridInstance.addEventListener("cellPress", listenerObject)
```

Description

Evénement diffusé à tous les objets écouteurs enregistrés lorsqu'un utilisateur clique dans une cellule.

Les composants utilisent un modèle dispatcher(diffuseur)/écouteur d'événement. Lorsqu'un composant DataGrid diffuse un événement `cellPress`, ce dernier est géré par une fonction (également appelée *gestionnaire*) associée à un objet écouteur (*`listenerObject`*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`eventObject`*) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. L'objet événement de l'événement `DataGrid.cellPress` possède trois propriétés supplémentaires :

`columnIndex` Nombre indiquant l'index de la colonne dans laquelle l'utilisateur a cliqué. La première position est 0.

`itemIndex` Nombre indiquant l'index de la ligne dans laquelle l'utilisateur a cliqué. La première position est 0.

`type` Chaîne `"cellPress"`.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `dgListener` est défini, puis transmis à la méthode `grid.addEventListener()` en tant que second paramètre. L'objet événement est capturé par le gestionnaire `cellPress` dans le paramètre *`evt_obj`*. Lorsque l'événement `cellPress` est diffusé, une instruction `trace` est envoyée au panneau Sortie. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` se trouve sur la scène, collez le code suivant dans la première image du scénario principal :

```
// Configuration des exemples de données.
my_dg.dataProvider = [{name:"Clark", score:3135}, {name:"Bruce",
  score:403}, {name:"Peter", score:25}];

// Création d'un objet écouteur.
var dgListener:Object = new Object();
dgListener.cellPress = function(evt_obj:Object) {
 var cell_str:String = "("+evt_obj.columnIndex+", "+evt_obj.itemIndex+")";
 trace("The cell at "+cell_str+" has been clicked");
};

// Ajout de l'écouteur.
my_dg.addEventListener("cellPress", dgListener);
```

DataGrid.change

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
listenerObject = new Object();
listenerObject.change = function(eventObject){
  // Insertion du code ici.
}
myDataGridInstance.addEventListener("change", listenerObject)
```

Description

Evénement diffusé à tous les objets écouteurs enregistrés lorsqu'un élément a été sélectionné.

Les composants utilisent un modèle dispatcher(diffuseur)/écouteur d'événement. Lorsqu'un composant DataGrid distribue un événement `change`, cet événement est géré par une fonction (également appelée *gestionnaire*) associée à un objet écouteur (*`listenerObject`*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. L'objet événement de l'événement `DataGrid.change` comprend une propriété supplémentaire, `type`, dont la valeur est `"change"`. Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `dgListener` est défini, puis transmis à la méthode `grid.addEventListener()` en tant que second paramètre. L'objet événement est capturé par le gestionnaire `change` dans le paramètre *evt_obj*. Lorsque l'événement `change` est diffusé, une instruction `trace` est envoyée au panneau Sortie. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` se trouve sur la scène, collez le code suivant dans la première image du scénario principal :

```
// Configuration des exemples de données.
my_dg.dataProvider = [{name:"Clark", score:3135}, {name:"Bruce",
  score:403}, {name:"Peter", score:25}];

// Création d'un objet écouteur.
var dgListener:Object = new Object();
dgListener.change = function(evt_obj:Object) {
 trace("The selection has changed to " + evt_obj.target.selectedIndex);
};

// Ajout de l'écouteur.
my_dg.addEventListener("change", dgListener);
```

DataGrid.columnCount

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid`.columnCount`

Description

Propriété (lecture seule) : nombre de colonnes affichées.

Exemple

L'exemple suivant affiche le nombre total de colonnes dans le panneau Sortie. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` se trouve sur la scène, collez le code suivant dans la première image du scénario principal :

```
// Ajout de colonnes à la grille et ajout de données.
my_dg.addColumn("a");
my_dg.addColumn("b");

my_dg.addItem({a:"one", b:"two"});

// Obtention du nombre de colonnes dans la grille.
var colCount_num:Number = my_dg.columnCount;
trace("Number of columns: "+colCount_num);
```

DataGrid.columnNames

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.columnNames

Description

Propriété : tableau des noms de champs (affichés sous forme de colonnes) de chaque élément.

Exemple

L'exemple suivant affiche le nom de la colonne dans le panneau Sortie lorsque vous cliquez sur le titre. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` se trouve sur la scène, collez le code suivant dans la première image du scénario principal :

```
my_dg.setSize(200, 100);
my_dg.columnNames = ["Name", "Description", "Price"];

var dgListener:Object = new Object();
dgListener.headerRelease = function (evt_obj:Object) {
 trace("You clicked on the \"" + my_dg.columnNames[evt_obj.columnIndex] +
   "\" column.");
}
my_dg.addEventListener("headerRelease", dgListener);
```

DataGrid.columnStretch

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
listenerObject = new Object();
listenerObject.columnStretch = function(eventObject){
  // Insertion du code ici.
}
myDataGridInstance.addEventListener("columnStretch", listenerObject)
```

Description

Evénement diffusé à l'ensemble des écouteurs enregistrés lorsque un utilisateur redimensionne une colonne horizontalement.

Les composants utilisent un modèle dispatcher(diffuseur)/écouteur d'événement. Lorsqu'un composant DataGrid distribue un événement `columnStretch`, cet événement est géré par une fonction (également appelée *gestionnaire*) associée à un objet écouteur (*`listenerObject`*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`eventObject`*) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. L'objet événement de l'événement `DataGrid.columnStretch` possède deux propriétés supplémentaires :

`columnIndex` Nombre indiquant l'index de la colonne cible. La première position est 0.

`type` Chaîne `"columnStretch"`.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

L'exemple suivant affiche le numéro d'index de la colonne dans le panneau Sortie lorsque vous redimensionnez le titre. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` se trouve sur la scène, collez le code suivant dans la première image du scénario principal :

```
my_dg.setSize(240, 100);

// Configuration des exemples de données.
var myDP_array:Array = new Array();
myDP_array.push({id:0, name:"Clark", score:3135});
myDP_array.push({id:1, name:"Bruce", score:403});
myDP_array.push({id:2, name:"Peter", score:25});

my_dg.dataProvider = myDP_array;

// Création d'un objet écouteur.
var dgListener:Object = new Object();
dgListener.columnStretch = function(evt_obj:Object) {
    trace("column " + evt_obj.columnIndex + " was resized");
};

// Ajout de l'écouteur.
my_dg.addEventListener("columnStretch", dgListener);
```

DataGrid.dataProvider

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.dataProvider

Description

Propriété : modèle de données des éléments affichés dans un composant DataGrid.

La grille de données ajoute des méthodes au prototype de la classe Array de sorte que chaque objet Array soit conforme à l'API DataProvider (voir le fichier DataProvider.as dans le dossier Classes/mx/controls/listclasses). Tout tableau existant dans la même image ou le même écran qu'une grille de données dispose automatiquement de toutes les méthodes (`addItem()`, `getItemAt()`, etc.) requises pour en faire le modèle de données d'une grille de données. Il peut être utilisé pour transmettre les modifications du modèle de données à plusieurs composants.

Dans un composant DataGrid, vous spécifiez les champs à afficher dans la propriété `DataGrid.columnNames`. Si vous ne définissez pas l'ensemble de colonnes (en définissant la propriété `DataGrid.columnNames` ou en appelant la méthode `DataGrid.addColumn()`) de la grille de données avant que la propriété `DataGrid.dataProvider` ne soit définie, la grille de données génère des colonnes pour chaque champ dans le premier élément du fournisseur de données, une fois cet élément généré.

Tout objet implémentant l'API DataProvider peut être utilisé en tant que fournisseur de données pour une grille de données (y compris les jeux d'enregistrements, les ensembles de données et les tableaux Flash Remoting). Pour consulter un exemple, reportez-vous à « DataSet.dataProvider », à la page 392.

Pour communiquer avec les données de la grille, servez-vous d'un fournisseur de données car ce dernier demeure cohérent, quel que soit l'emplacement de défilement.

Exemple

L'exemple suivant permet de créer un tableau à utiliser comme fournisseur de données et de l'assigner directement à la propriété `dataProvider` :

```
my_dg.dataProvider = [{name:"Chris", price:"Priceless"}, {name:"Nigel",
  price:"cheap"}];
```

L'exemple suivant crée un nouvel objet Array (tableau) décoré avec l'API DataProvider. Il utilise une boucle `for` pour ajouter 20 éléments à la grille :

```
var myDP:Array = new Array();
for (var i=0; i<20; i++)
  myDP.addItem({id:i, name:"Dave", price:"Priceless"});
my_dg.dataProvider = myDP
```

DataGrid.editable

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid`.editable`

Description

Propriété : détermine si la grille de données peut être modifiée par l'utilisateur (`true`) ou non (`false`). Pour que les colonnes puissent être modifiées individuellement et que les cellules puissent recevoir le focus, cette propriété doit avoir la valeur `true`. La valeur par défaut est `false`.

Si vous voulez que certaines colonnes ne soient pas modifiables, utilisez la propriété `DataGridColumn.editable`.

ATTENTION	La grille de données n'est pas modifiable et ne peut pas être triée lorsqu'elle est liée directement à un composant WebServiceConnector ou XMLConnector. Pour que la grille puisse être modifiée ou triée, liez le composant DataGrid au composant DataSet, puis ce dernier au composant WebServiceConnector ou XMLConnector.

Exemple

L'exemple suivant permet de modifier toutes les colonnes de la grille, sauf la première. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` se trouve sur la scène, collez le code suivant dans la première image du scénario principal :

```
my_dg.setSize(140, 100);

// Ajout de colonnes à la grille et ajout de données.
my_dg.addColumn("a");
my_dg.addColumn("b");
my_dg.addItem({a:"one", b:1});
my_dg.addItem({a:"two", b:2});

// Transformation du composant DataGrid en composant modifiable.
my_dg.editable = true;
// Transformation de la première colonne en une colonne en lecture seule.
my_dg.getColumnAt(0).editable = false;
```

Voir aussi

`DataGridColumn.editable`

DataGrid.editField()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.editField(*index*, *colName*, *data*)

Paramètres

index Index de la cellule cible. La numérotation de cette valeur commence à zéro.

colName Chaîne indiquant le nom de la colonne (champ) qui contient la cellule cible.

data Valeur à stocker dans la cellule cible. Ce paramètre peut être de tout type.

Valeur renvoyée

Les données qui étaient dans la cellule.

Description

Méthode qui remplace les données de la cellule à l'emplacement spécifié et actualise la grille de données avec la nouvelle valeur. La méthode `setValue()` est déclenchée pour toute cellule présente pour cette valeur.

Exemple

L'exemple suivant place une valeur dans la grille sur la première ligne de la première colonne (valeur d'indexation `0`) lorsque vous cliquez sur le bouton. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` et une occurrence du composant Button nommée `my_btn` se trouvent sur la scène, collez le code suivant dans la première image du scénario principal :

```
my_dg.setSize(140, 100);

// Configuration des exemples de données.
my_dg.dataProvider = [{name:"Clark", score:3135}, {name:"Bruce",
  score:403}, {name:"Peter", score:25}];

// Création d'un objet écouteur.
var btnListener:Object = new Object();
btnListener.click = function() {
   // Remplacement du premier champ par de nouvelles valeurs.
   my_dg.editField(0, "name", "Arthur");
};

// Ajout de l'écouteur du composant Button.
my_btn.addEventListener("click", btnListener);
```

DataGrid.focusedCell

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.focusedCell

Description

Propriété : en mode d'édition uniquement, occurrence d'objet définissant la cellule qui a le focus. L'objet doit avoir les champs `columnIndex` et `itemIndex`, tous deux des nombres entiers indiquant l'index de la colonne et l'élément de la cellule. L'origine est (0,0). La valeur par défaut est `undefined`.

Exemple

L'exemple suivant définit la cellule ayant le focus sur la deuxième colonne, onzième ligne (numérotée « 10 », car la première ligne est la « 0 »). Comme vous ne pouvez pas accéder aux cellules tant que le composant DataGrid n'est pas dessiné, utilisez `UIObject.doLater()` pour différer l'utilisation de la propriété `focusedCell` :

```
// Création d'un fournisseur de données avec trois colonnes et 50 lignes.
var myDP:Array = new Array();
for (var i=0; i<50; i++)
  myDP.addItem({id:i, name:"Dave", price:"Priceless"});

// Affectation du fournisseur de données à l'occurrence DataGrid et
// transformation de cette occurrence en grille modifiable.
my_dg.dataProvider = myDP;
my_dg.editable = true;

// Utilisation de UIObject.doLater() dans le scénario actuel pour appeler
// la fonction dès que la grille de données a défini l'ensemble
// de ses propriétés.
my_dg.doLater(this, "select");

function select() {
 my_dg.focusedCell = {columnIndex:1, itemIndex:10};
}
```

DataGrid.getColumnAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.getColumnAt(*index*)

Paramètres

index Index de l'objet DataGridColumn à renvoyer. La numérotation de cette valeur commence à zéro.

Valeur renvoyée

Un objet DataGridColumn.

Description

Méthode qui obtient une référence à l'objet DataGridColumn au niveau de l'index spécifié.

Exemple

L'exemple suivant obtient l'objet DataGridColumn à l'index 0 et modifie le texte. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` et une occurrence du composant Button nommée `my_btn` se trouvent sur la scène, collez le code suivant dans la première image du scénario principal :

```
my_dg.setSize(140, 100);

// Configuration des exemples de données.
my_dg.dataProvider = [{name:"Clark", score:3135}, {name:"Bruce",
  score:403}, {name:"Peter", score:25}];

// Création d'un objet écouteur.
var btnListener:Object = new Object();
btnListener.click = function() {
  // Obtention de la colonne à l'emplacement 0.
  var a_dgc = my_dg.getColumnAt(0);
  // Modification du texte de l'en-tête.
  a_dgc.headerText = "c";
};

// Ajout de l'écouteur du composant Button.
my_btn.addEventListener("click", btnListener);
```

DataGrid.getColumnIndex()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.getColumnIndex(*columnName*)

Paramètres

columnName Chaîne correspondant au nom d'une colonne.

Valeur renvoyée

Un nombre indiquant l'index de la colonne.

Description

Méthode : renvoie l'index de la colonne spécifiée par le paramètre *columnName*.

Exemple

L'exemple suivant affiche le numéro d'index de la colonne « score ». Lorsqu'une occurrence du composant DataGrid appelée my_dg se trouve sur la scène, collez le code suivant dans la première image du scénario principal :

```
my_dg.setSize(150, 100);

// Configuration des exemples de données.
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});

my_dg.dataProvider = myDP_array;

var column_num:Number = my_dg.getColumnIndex("score");
trace("Column that has name of 'score': " + column_num);
```

DataGrid.headerHeight

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.headerHeight

Description

Propriété : hauteur de la barre d'en-tête de la grille de données, en pixels. La valeur par défaut est 20.

Exemple

L'exemple suivant définit la hauteur de la barre d'en-tête sur 40. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` se trouve sur la scène, collez le code suivant dans la première image du scénario principal :

```
// Définition des attributs de la grille.
my_dg.setSize(240, 100);
my_dg.spaceColumnsEqually();

// Configuration des exemples de données.
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});
my_dg.dataProvider = myDP_array;

my_dg.headerHeight = 40;
```

DataGrid.headerRelease

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
listenerObject = new Object();
listenerObject.headerRelease = function(eventObject){
  // Insertion du code ici.
}
myDataGridInstance.addEventListener("headerRelease", listenerObject)
```

Description

Evénement diffusé à tous les écouteurs enregistrés lorsque l'utilisateur désélectionne un en-tête de colonne. Vous pouvez l'utiliser avec la propriété `DataGridColumn.sortOnHeaderRelease` pour empêcher tout tri automatique et vous permettre d'effectuer le tri selon vos préférences.

Les composants utilisent un modÃ¨le d'événement dispatcher(diffuseur)/écouteur. Lorsque le composant DataGrid distribue un événement `headerRelease`, cet événement est géré par une fonction (également appelée *gestionnaire*) associée à un objet écouteur (*`listenerObject`*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`eventObject`*) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. L'objet événement de l'événement `DataGrid.headerRelease` possède deux propriétés supplémentaires :

`columnIndex` Nombre indiquant l'index de la colonne cible.

`type` Chaîne `"headerRelease"`.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `myListener` est défini, puis transmis à la méthode `grid.addEventListener()` en tant que second paramètre. L'objet événement est capturé par le gestionnaire `headerRelease` dans le paramètre *`eventObject`*. Lorsque l'événement `headerRelease` est diffusé, une instruction `trace` est envoyée au panneau Sortie.

```
var myListener = new Object();
myListener.headerRelease = function(event) {
    trace("column " + event.columnIndex + " header was pressed");
};
grid.addEventListener("headerRelease", myListener);
```

Dans l'exemple suivant, vous modifiez l'ordre de tri à l'aide d'une colonne. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` se trouve sur la scène, collez le code suivant dans la première image du scénario principal :

```
var my_dg:mx.controls.DataGrid;
my_dg.setSize(150, 100);
my_dg.spaceColumnsEqually();
var myListener:Object = new Object();
myListener.headerRelease = function(evt:Object) {
  trace("column "+evt.columnIndex+" header was pressed");
  trace("\t current sort order is: "+evt.target.sortDirection);
  trace("");
};
my_dg.addEventListener("headerRelease", myListener);

my_dg.addColumn("a");
my_dg.addColumn("b");
my_dg.addItem({a:'one', b:1});
my_dg.addItem({a:'two', b:2});
```

En accédant à la propriété `sortDirection`, vous pouvez alors définir un ordre de tri croissant ou décroissant. La propriété `sortDirection` étant une chaîne, elle recherche soit `ASC`, soit `DESC`.

DataGrid.hScrollPolicy

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`myDataGrid``.hScrollPolicy`

Description

Propriété qui spécifie si la grille de données comprend une barre de défilement horizontale. La valeur de cette propriété est `"on"`, `"off"` ou `"auto"`. La valeur par défaut est `"off"`.

Si vous avez défini `hScrollPolicy` sur `"off"`, les colonnes sont redimensionnées proportionnellement pour être adaptées à la largeur déterminée.

REMARQUE	Ce n'est pas le cas pour le composant List dont la propriété `hScrollPolicy` ne peut pas être définie sur `"auto"`.

Exemple

L'exemple suivant définit une règle de défilement horizontal automatique, c'est-à-dire qu'une barre de défilement horizontale apparaît lorsque tout le contenu doit être affiché :

```
my_dg.setSize(150, 100);

// Ajout de colonnes à la grille et ajout de données.
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});

my_dg.dataProvider = myDP_array;

my_dg.hScrollPolicy = "on";
```

DataGrid.removeAllColumns()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid`.removeAllColumns()`

Paramètres

Aucun.

Valeur renvoyée

Aucune.

Description

Méthode qui supprime tous les objets DataGridColumn de la grille de données. Son appel n'a aucun effet sur le fournisseur de données.

Appelez-la si vous définissez un nouveau fournisseur de données dont les champs diffèrent du précédent et lorsque vous souhaitez effacer les champs affichés.

Exemple

L'exemple suivant supprime tous les objets DataGridColumn du composant DataGrid lorsque vous cliquez sur le bouton. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` et une occurrence du composant Button nommée `my_btn` se trouvent sur la scène, collez le code suivant dans la première image du scénario principal :

```
my_dg.setSize(140, 100);
my_dg.move(10, 40);

this.createClassObject(mx.controls.Button, "clear_button", 20,
  {label:"Clear"});
clear_button.move(10, 10);

// Configuration des exemples de données.
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});
my_dg.dataProvider = myDP_array;

var buttonListener:Object = new Object();
buttonListener.click = function (evt_obj:Object) {
 my_dg.removeAllColumns();
}
clear_button.addEventListener("click", buttonListener);
```

DataGrid.removeColumnAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.removeColumnAt(*index*)

Paramètres

index Index de la colonne à supprimer.

Valeur renvoyée

Une référence à l'objet DataGridColumn supprimé.

Description

Méthode qui supprime l'objet DataGridColumn à l'index spécifié.

Exemple

L'exemple suivant supprime le premier objet DataGridColumn lorsque vous cliquez sur le bouton. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` et une occurrence du composant Button nommée `name_button` se trouvent sur la scène, collez le code suivant dans la première image du scénario principal :

```
my_dg.setSize(140, 100);
my_dg.move(10, 40);

name_button.setSize(140, name_button.height);
name_button.move(10, 10);

// Configuration des exemples de données.
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});
my_dg.dataProvider = myDP_array;

// Création d'un objet écouteur.
var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object) {
 my_dg.removeColumnAt(my_dg.getColumnIndex("name"));
 evt_obj.target.enabled = false;
};
// Ajout de l'écouteur du composant Button.
name_button.addEventListener("click", buttonListener);
```

DataGrid.replaceItemAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.replaceItemAt(*index*, *item*)

Paramètres

index Index de l'élément à remplacer.

item Objet correspondant à la valeur de l'élément à utiliser comme valeur de remplacement.

Valeur renvoyée

La valeur précédente.

Description

Méthode qui remplace l'élément à l'index spécifié et actualise l'affichage de la grille.

Exemple

L'exemple suivant remplace l'élément à l'index de ligne 2 par de nouvelles entrées. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` et une occurrence du composant Button nommée `replace_button` se trouvent sur la scène, collez le code suivant dans la première image du scénario principal :

```
my_dg.setSize(140, 100);
my_dg.move(10, 40);

replace_button.move(10, 10);

// Configuration des exemples de données.
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});
my_dg.dataProvider = myDP_array;
```

```
// Création d'un objet écouteur.
var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object) {
 //Remplacement de la valeur précédente
 var prevValue_obj:Object = my_dg.replaceItemAt(2, {name:"Frank",
  score:949});
 my_dg.selectedIndex = 2;
};
// Ajout de l'écouteur du composant Button.
replace_button.addEventListener("click", buttonListener);
```

DataGrid.resizableColumns

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.resizableColumns

Description

Propriété : valeur booléenne indiquant si les colonnes de la grille peuvent être étirées (`true`) ou non (`false`) par l'utilisateur. Pour que les colonnes individuelles puissent être redimensionnées, cette propriété doit être définie sur `true`. La valeur par défaut est `true`.

Exemple

L'exemple suivant permet d'empêcher les utilisateurs de redimensionner les colonnes. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` se trouve sur la scène, collez le code suivant dans la première image du scénario principal :

```
my_dg.setSize(140, 100);

// Configuration des exemples de données.
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});
my_dg.dataProvider = myDP_array;

// Interdiction du redimensionnement des colonnes.
my_dg.resizableColumns = false;
```

DataGrid.selectable

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.selectable

Description

Propriété : valeur booléenne indiquant si l'utilisateur peut sélectionner (`true`) ou non (`false`) la grille de données. La valeur par défaut est `true`. Si la valeur est `false`, un élément de la grille ne reste pas sélectionné une fois que vous avez cliqué sur l'élément et déplacé le curseur.

Exemple

L'exemple suivant empêche l'utilisateur de sélectionner la grille. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` se trouve sur la scène, collez le code suivant dans la première image du scénario principal :

```
my_dg.setSize(140, 100);

// Configuration des exemples de données.
var myDP_array:Array = new Array();
myDP_array.push({name:"Clark", score:3135});
myDP_array.push({name:"Bruce", score:403});
myDP_array.push({name:"Peter", score:25});
my_dg.dataProvider = myDP_array;

my_dg.selectable = false;
```

DataGrid.showHeaders

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.showHeaders

Description

Propriété : valeur booléenne indiquant si la grille de données doit afficher (`true`) ou non (`false`) les en-têtes de colonnes. Pour mieux être distingués des autres lignes d'une grille, les en-têtes de colonnes sont grisés. Si `DataGrid.sortableColumns` est défini sur `true`, l'utilisateur peut trier le contenu d'une colonne en cliquant sur son en-tête. La valeur par défaut de `showHeaders` est `true`.

Exemple

L'exemple suivant masque les en-têtes de colonnes :

```
my_dg.setSize(140, 100);

// Configuration des exemples de données.
my_dg.addItem({name:"Clark", score:3135});
my_dg.addItem({name:"Bruce", score:403});
my_dg.addItem({name:"Peter", score:25});

// Pas d'affichage des en-têtes.
my_dg.showHeaders = false;
```

Voir aussi

`DataGrid.sortableColumns`

DataGrid.sortableColumns

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid`.sortableColumns`

Description

Propriété : valeur booléenne indiquant si les colonnes de la grille de données peuvent être triées (`true`) ou non (`false`) lorsque l'utilisateur clique sur leurs en-têtes. Cette propriété doit être définie sur `true` pour que les colonnes individuelles puissent être triées et l'événement `headerRelease` diffusé. La valeur par défaut est `true`.

ATTENTION	La grille de données n'est pas modifiable et ne peut pas être triée lorsqu'elle est liée directement à un composant WebServiceConnector ou XMLConnector. Pour que la grille puisse être modifiée ou triée, liez le composant DataGrid au composant DataSet, puis ce dernier au composant WebServiceConnector ou XMLConnector.

Exemple

L'exemple suivant désactive le tri :

```
my_dg.setSize(140, 100);

// Configuration des exemples de données.
my_dg.addItem({name:"Clark", score:3135});
my_dg.addItem({name:"Bruce", score:403});
my_dg.addItem({name:"Peter", score:25});

// Interdiction du tri des colonnes.
my_dg.sortableColumns = false;
```

Voir aussi

`DataGrid.headerRelease`

DataGrid.spaceColumnsEqually()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid`.spaceColumnsEqually()`

Paramètres

Aucun.

Valeur renvoyée

Aucune.

Description

Méthode qui redimensionne les colonnes de façon égale.

Exemple

L'exemple suivant redimensionne les colonnes de `my_dg` lorsque vous cliquez sur le bouton. Lorsqu'une occurrence du composant DataGrid appelée `my_dg` et une occurrence du composant Button nommée `resize_button` se trouvent sur la scène, collez le code suivant dans la première image du scénario principal :

```
my_dg.move(10, 40);
my_dg.setSize(200, 100);

resize_button.move(10, 10);
resize_button.setSize(200, resize_button.height);

my_dg.addColumn("guitar");
my_dg.addColumn("name");

// Configuration des exemples de données.
my_dg.addItem({guitar:"Flying V", name:"maggot"});
my_dg.addItem({guitar:"SG", name:"dreschie"});
my_dg.addItem({guitar:"jagstang", name:"vitapup"});

// Création d'un objet écouteur.
var buttonListener:Object = new Object();
buttonListener.click = function() {
 my_dg.spaceColumnsEqually();
};
// Ajout de l'écouteur du composant Button.
resize_button.addEventListener("click", buttonListener);
```

Classe DataGridColumn

Nom de classe ActionScript mx.controls.gridclasses.DataGridColumn

Vous pouvez créer et configurer des objets DataGridColumn à utiliser en tant que colonnes d'une grille de données. Un grand nombre de méthodes de la classe DataGrid est consacré à la gestion des objets DataGridColumn. Dans la grille de données, les objets DataGridColumn sont triés dans un tableau basé sur zéro, c'est-à-dire la colonne la plus à gauche. Une fois les colonnes ajoutées ou créées, vous pouvez appeler `DataGrid.getColumnAt(`*`index`*`)` pour y accéder.

Il existe trois manières d'ajouter ou de créer des colonnes dans une grille. Si vous souhaitez configurer les colonnes, il est préférable d'utiliser la deuxième ou la troisième méthode avant d'ajouter des données dans une grille. Ainsi vous n'aurez pas à créer les colonnes deux fois.

- Ajoutez un fournisseur de données ou un élément comportant plusieurs champs à une grille qui ne possède aucun objet DataGridColumn configuré. Cette méthode génère automatiquement des colonnes pour tous les champs, dans l'ordre inverse de la boucle `for..in`. Par exemple, pour une occurrence du composant DataGrid appelée `my_dg` :

```
my_dg.dataProvider = [{guitar:"Flying V", name:"maggot"}, {guitar:"SG",
  name:"dreschie"}, {guitar:"jagstang", name:"vitapup"}];
```

- Utilisez `DataGrid.columnNames` pour créer les noms des champs d'éléments souhaités et générer des objets DataGridColumn, dans l'ordre, pour chaque champ répertorié. Cette approche vous permet de sélectionner et de classer rapidement les colonnes, en faisant appel à un minimum de paramètres de configuration. Elle supprime toute information précédente liée à la colonne. Par exemple, pour une occurrence du composant DataGrid appelée `my_dg` :

```
my_dg.columnNames = ["guitar","name"];
```

- Précréez les objets DataGridColumn et ajoutez-les à la grille de données en utilisant `DataGrid.addColumn()`. Cette approche est très pratique, car elle permet d'ajouter des colonnes de taille et de format adéquats alors qu'elles ne se trouvent pas encore dans la grille (le processeur est moins sollicité). Pour plus d'informations, voir « Constructeur de la classe DataGridColumn », à la page 317. Par exemple, pour une occurrence du composant DataGrid appelée `my_dg` :

```
// Création d'un objet de colonne.
var location_dgc:DataGridColumn = new DataGridColumn("Location");
location_dgc.width = 100;
// Ajout de la colonne au composant DataGrid.
my_dg.addColumn(location_dgc);
```

Propriétés de la classe DataGridColumn

Le tableau suivant présente les propriétés de la classe DataGridColumn.

Propriété	Description
`DataGridColumn.cellRenderer`	Identifiant de liaison d'un symbole à utiliser pour afficher les cellules dans cette colonne.
`DataGridColumn.columnName`	Lecture seule ; nom du champ associé à la colonne.
`DataGridColumn.editable`	Une valeur booléenne indiquant si une colonne est modifiable (`true`) ou non (`false`).

Propriété	Description
DataGridColumn.headerRenderer	Nom de la classe à utiliser pour afficher l'en-tête de cette colonne.
DataGridColumn.headerText	Texte de l'en-tête de cette colonne.
DataGridColumn.labelFunction	Fonction qui détermine le champ d'un élément à afficher.
DataGridColumn.resizable	Une valeur booléenne indiquant si une colonne peut être redimensionnée (true) ou non (false).
DataGridColumn.sortable	Une valeur booléenne indiquant si une colonne peut être triée (true) ou non (false).
DataGridColumn.sortOnHeaderRelease	Valeur booléenne indiquant si une colonne peut être triée (true) ou non (false) lorsque l'utilisateur clique sur son en-tête.
DataGridColumn.width	Largeur d'une colonne, en pixels.

Constructeur de la classe DataGridColumn

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
new DataGridColumn(name)
```

Paramètres

name Chaîne indiquant le nom de l'objet DataGridColumn. Ce paramètre correspond au champ de chaque élément à afficher.

Valeur renvoyée

Aucune.

Description

Constructeur qui crée un objet DataGridColumn. Utilisez-le pour créer des colonnes à ajouter à un composant DataGrid. Une fois les objets DataGridColumn créés, vous pouvez les ajouter à une grille de données en appelant DataGrid.addColumn().

Exemple

L'exemple suivant crée un objet DataGridColumn appelé `Location` :

```
import mx.controls.gridclasses.DataGridColumn;
var column = new DataGridColumn("Location");
```

DataGridColumn.cellRenderer

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`myDataGrid``.getColumnAt(`*`index`*`).cellRenderer`

Description

Propriété : identifiant de liaison pour un symbole à utiliser pour afficher des cellules dans cette colonne. Toutes les classes utilisées pour cette propriété doivent implémenter l'API CellRenderer (voir « API CellRenderer », à la page 111.) La valeur par défaut est `undefined`.

Exemple

L'exemple suivant utilise un identifiant de liaison pour définir un nouveau rendu de cellule :

```
myGrid.getColumnAt(3).cellRenderer = "MyCellRenderer";
```

DataGridColumn.columnName

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`myDataGrid``.getColumnAt(`*`index`*`).columnName`

Description

Propriété (lecture seule) : nom du champ associé à cette colonne. La valeur par défaut est le nom dont il est question dans le constructeur DataGridColumn.

Exemple

L'exemple suivant affiche le nom de la colonne à la position d'index 1 :

```
import mx.controls.gridclasses.DataGridColumn;
// Définition des attributs de la grille.
my_dg.setSize(150, 100);

// Ajout de colonnes à la grille.
var name_dgc:DataGridColumn = my_dg.addColumn(new DataGridColumn("name"));
name_dgc.headerText = "Name:";
var score_dgc:DataGridColumn = my_dg.addColumn(new
  DataGridColumn("score"));
score_dgc.headerText = "Score:";

// Configuration des exemples de données.
my_dg.addItem({name:"Clark", score:3135});
my_dg.addItem({name:"Bruce", score:403});
my_dg.addItem({name:"Peter", score:25});

// Obtention du nom de la colonne.
var name_str:String = my_dg.getColumnAt(1).columnName;
trace(name_str);
```

Voir aussi

Constructeur de la classe DataGridColumn

DataGridColumn.editable

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.getColumnAt(*index*).editable

Description

Propriété : détermine si l'utilisateur peut modifier (`true`) ou non (`false`) la colonne. La propriété `DataGrid.editable` doit être définie sur `true` pour que les colonnes individuelles puissent être modifiées, même lorsque `DataGridColumn.editable` est défini sur `true`. La valeur par défaut est `true`.

ATTENTION	La grille de données n'est pas modifiable et ne peut pas être triée lorsqu'elle est liée directement à un composant WebServiceConnector ou XMLConnector. Pour que la grille puisse être modifiée ou triée, liez le composant DataGrid au composant DataSet, puis ce dernier au composant WebServiceConnector ou XMLConnector.

Exemple

L'exemple suivant rend les éléments de la première colonne d'une grille non modifiables :

```
// Définition des attributs de la grille.
my_dg.setSize(150, 100);
my_dg.editable = true;

// Ajout de colonnes à la grille.
my_dg.addColumn("name");
my_dg.addColumn("score");

// Configuration des exemples de données.
my_dg.addItem({name:"Clark", score:3135});
my_dg.addItem({name:"Bruce", score:403});
my_dg.addItem({name:"Peter", score:25});

// Interdiction de la modification de la première colonne.
my_dg.getColumnAt(0).editable = false;
```

Voir aussi

`DataGrid.editable`

DataGridColumn.headerRenderer

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`myDataGrid``.getColumnAt(`*`index`*`).headerRenderer`

Description

Propriété : chaîne indiquant le nom de classe à utiliser pour afficher l'en-tête de cette colonne. Toutes les classes utilisées pour cette propriété doivent implémenter l'API CellRenderer (voir « API CellRenderer », à la page 111). La valeur par défaut est `undefined`.

Exemple

L'exemple suivant utilise un identifiant de liaison pour définir un nouvel objet headerRenderer :

```
myGrid.getColumnAt(3).headerRenderer = "MyHeaderRenderer";
```

DataGridColumn.headerText

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.getColumnAt(*index*).headerText

Description

Propriété : texte de l'en-tête de colonne. La valeur par défaut est le nom de la colonne.

Cette propriété vous permet de choisir un autre en-tête que le nom du champ.

Exemple

L'exemple suivant définit l'en-tête de colonne sur « Price (USD) » :

```
import mx.controls.gridclasses.DataGridColumn;

var my_dg:mx.controls.DataGrid;

var price_dgc:DataGridColumn = new DataGridColumn("price");
price_dgc.headerText = "Price (USD)";
price_dgc.width = 80;
my_dg.addColumn(price_dgc);

my_dg.addItem({price:"$14.99"});
```

DataGridColumn.labelFunction

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.getColumnAt(*index*).labelFunction

Description

Propriété qui spécifie une fonction permettant de déterminer le champ (ou la combinaison de champs) à afficher pour chaque élément. Cette fonction reçoit un paramètre, *item*, qui correspond à l'élément restitué et doit renvoyer une chaîne représentant le texte à afficher. Cette propriété peut être utilisée pour créer des colonnes virtuelles qui n'ont pas de champ équivalent dans l'élément.

REMARQUE	La fonction spécifiée opère dans un domaine non défini.

Exemple

L'exemple suivant calcule une valeur pour la colonne « Subtotal » :

```
import mx.controls.gridclasses.DataGridColumn;

var my_dg:mx.controls.DataGrid;
my_dg.setSize(300, 200);

// Configuration des colonnes.
var guitar_dgc:DataGridColumn = new DataGridColumn("guitar");
var value_dgc:DataGridColumn = new DataGridColumn("value");
var tax_dgc:DataGridColumn = new DataGridColumn("tax");
var st_dgc:DataGridColumn = new DataGridColumn("Subtotal");
// Définition de labelFunction pour la colonne Subtotal.
st_dgc.labelFunction = function(item:Object):String {
 if ((item.value != undefined) && (item.tax != undefined)) {
  return "$"+(item.value+item.tax);
 }
};

// Ajout de colonnes à la grille.
my_dg.addColumn(guitar_dgc);
my_dg.addColumn(value_dgc);
my_dg.addColumn(tax_dgc);
my_dg.addColumn(st_dgc);

// Définition du modèle de données.
my_dg.addItem({guitar:"Flying V", value:10, tax:1});
my_dg.addItem({guitar:"SG", value:20, tax:2});
my_dg.addItem({guitar:"jagstang", value:30, tax:3});
```

DataGridColumn.resizable

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.getColumnAt(*index*).resizable

Description

Propriété : valeur booléenne indiquant si l'utilisateur peut redimensionner (`true`) ou non (`false`) une colonne. La propriété `DataGrid.resizableColumns` doit être définie sur `true` pour qu'elle prenne effet. La valeur par défaut est `true`.

Exemple

L'exemple suivant empêche l'utilisateur de redimensionner la colonne à l'index 0 :

```
// Définition des attributs de la grille.
my_dg.setSize(150, 100);
my_dg.addColumn("name");
my_dg.addColumn("score");

// Configuration des exemples de données.
my_dg.addItem({name:"Clark", score:3135});
my_dg.addItem({name:"Bruce", score:403});
my_dg.addItem({name:"Peter", score:25});

// Interdiction du redimensionnement de la première colonne.
my_dg.getColumnAt(0).resizable = false;
```

DataGridColumn.sortable

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.getColumnAt(*index*).sortable

Description

Propriété : valeur booléenne indiquant si l'utilisateur peut trier (`true`) ou non (`false`) une colonne. La propriété `DataGrid.sortableColumns` doit être définie sur `true` pour qu'elle prenne effet. La valeur par défaut est `true`.

ATTENTION	La grille de données n'est pas modifiable et ne peut pas être triée lorsqu'elle est liée directement à un composant WebServiceConnector ou XMLConnector. Pour que la grille puisse être modifiée ou triée, liez le composant DataGrid au composant DataSet, puis ce dernier au composant WebServiceConnector ou XMLConnector.

Exemple

L'exemple suivant empêche l'utilisateur de trier la colonne à l'index 1 :

```
// Définition des attributs de la grille.
my_dg.setSize(150, 100);

// Configuration des exemples de données.
my_dg.addItem({name:"Clark", score:3135});
my_dg.addItem({name:"Bruce", score:403});
my_dg.addItem({name:"Peter", score:25});

// Interdiction du tri de la deuxième colonne.
my_dg.getColumnAt(1).sortable = false;
```

DataGridColumn.sortOnHeaderRelease

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.getColumnAt(*index*).sortOnHeaderRelease

Description

Propriété : valeur booléenne indiquant si la colonne peut être triée automatiquement (`true`) ou non (`false`) lorsque l'utilisateur clique sur un en-tête. Cette propriété ne peut être définie sur `true` que si `DataGridColumn.sortable` est définie sur `true` également.
Si `DataGridColumn.sortOnHeaderRelease` est défini sur `false`, vous pouvez utiliser l'événement `headerRelease` et effectuer votre propre tri.

La valeur par défaut est `true`.

ATTENTION	La grille de données n'est pas modifiable et ne peut pas être triée lorsqu'elle est liée directement à un composant WebServiceConnector ou XMLConnector. Pour que la grille puisse être modifiée ou triée, liez le composant DataGrid au composant DataSet, puis ce dernier au composant WebServiceConnector ou XMLConnector.

Exemple

L'exemple suivant désactive le tri de la deuxième colonne :

```
// Définition des attributs de la grille.
my_dg.setSize(150, 100);

// Configuration des exemples de données.
my_dg.addItem({name:"Clark", score:3135});
my_dg.addItem({name:"Bruce", score:403});
my_dg.addItem({name:"Peter", score:25});

// Ne pas trier la deuxième colonne en cliquant sur l'en-tête.
my_dg.getColumnAt(1).sortOnHeaderRelease = false;
```

DataGridColumn.width

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDataGrid.getColumnAt(*index*).width

Description

Propriété : nombre qui indique la largeur d'une colonne, en pixels. La valeur par défaut est 50.

Exemple

L'exemple suivant définit la largeur de la première colonne sur 50 pixels :

```
// Création d'un composant DataProvider.
var myDP_array:Array = new Array({name:"Chris", price:"Priceless"},
  {name:"Nigel", price:"Cheap"});
// Affectation du composant DataProvider à un composant DataGrid.
my_dg.dataProvider = myDP_array;

// Modification des dimensions du composant DataGrid.
my_dg.setSize(140, 100);
my_dg.rowHeight = 30;
my_dg.getColumnAt(0).width = 50;
```

CHAPITRE 11

Composant DataHolder

11

Le composant DataHolder est un référentiel pour les données et un moyen de générer des événements lors de la modification de ces données. Il permet essentiellement de stocker les données et joue le rôle de connecteur entre d'autres composants qui utilisent la liaison de données.

REMARQUE

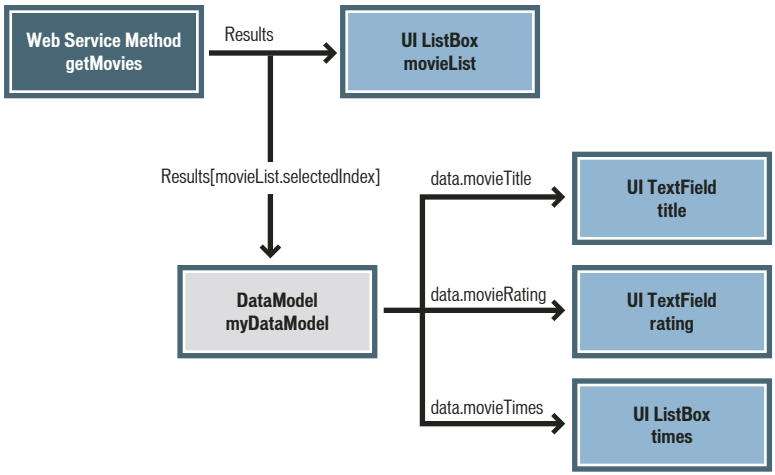
Le composant DataHolder est pris en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Initialement, le composant DataHolder possède une seule propriété de liaison, appelée `data`. L'onglet Schéma de l'inspecteur de composants vous permet de lui en ajouter d'autres. Pour plus d'informations sur l'utilisation de l'onglet Schéma, reportez-vous à *Utilisation des schémas dans l'onglet Schéma* du manuel *Utilisation de Flash*.

Vous pouvez affecter tout type de données à un composant DataHolder, soit en créant une liaison entre les données et une autre propriété, soit en utilisant votre propre code ActionScript. Lorsque les valeurs des données sont modifiées, le composant DataHolder émet un événement dont le nom est identique à celui de la propriété et exécute toute liaison associée à cette propriété.

Dans la plupart des cas, vous n'utilisez pas ce composant pour construire une application. Il n'est requis que lorsque vous ne pouvez pas lier directement des données externes à un autre composant et que vous ne souhaitez pas utiliser un composant DataSet. Le composant DataHolder est particulièrement utile lorsqu'il est impossible d'associer directement des composants (tels que des connecteurs, des composants d'interface utilisateur ou des composants DataSet). Vous trouverez ci-dessous des scénarios dans lesquels vous pouvez utiliser un composant DataHolder :

- Si une valeur de données est générée par ActionScript, vous pouvez l'associer à d'autres composants. Dans ce cas, vous pouvez avoir un composant DataHolder qui contient des propriétés associées à votre convenance. Lorsque de nouvelles valeurs sont affectées à ces propriétés (à l'aide d'ActionScript, par exemple), ces valeurs sont distribuées à l'objet de liaison des données.
- Vous pouvez par exemple avoir une valeur de données résultant d'une liaison des données indexée et complexe, comme dans le diagramme suivant :



Dans ce cas, il est pratique de lier la valeur de données à un composant DataHolder (appelé *DataModel* dans cette illustration), puis de l'utiliser pour des liaisons avec l'interface utilisateur.

REMARQUE	Le composant DataHolder n'est pas destiné à implémenter le même contrôle sur vos données que le composant DataSet. Il est incapable de gérer, suivre ou mettre à jour des données. Il s'agit d'un référentiel permettant de stocker des données et de générer des événements lors de leur modification.

Création d'une application avec le composant DataHolder

Dans cet exemple, vous ajoutez une propriété Array à un schéma du composant DataHolder (un tableau) dont la valeur est déterminée par le code ActionScript que vous rédigez. Dans l'onglet Liaisons de l'inspecteur de composants, liez ensuite cette propriété Array à la propriété `dataProvider` d'un composant DataGrid.

Pour utiliser le composant DataHolder dans une application simple :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser un composant DataHolder du panneau Composants sur la scène et nommez-le **dataHolder**.
3. Faites glisser un composant DataGrid sur la scène et nommez-le **namesGrid**.
4. Sélectionnez le composant DataHolder et ouvrez l'inspecteur des composants.
5. Cliquez sur l'onglet Schéma de l'inspecteur des composants.
6. Cliquez sur le bouton Ajouter une propriété de composant (+), situé dans le panneau supérieur de l'onglet Schéma.
7. Dans le panneau inférieur de l'onglet Schéma, tapez **namesArray** dans le champ Nom du champ, puis sélectionnez Array dans le menu déroulant Type de données.
8. Dans l'inspecteur de composants, cliquez sur l'onglet Liaisons et ajoutez une liaison entre la propriété `namesArray` du composant DataHolder et la propriété `dataProvider` du composant DataGrid.

   Pour plus d'informations sur la création de liaisons via l'onglet Liaisons, reportez-vous à *Utilisation des liaisons dans l'onglet Liaisons* du manuel *Utilisation de Flash*.
9. Dans le scénario, sélectionnez la première image dans le calque 1 et ouvrez le panneau Actions.
10. Entrez le code suivant dans le panneau Actions :

    ```
    dataHolder.namesArray = [{name:"Tim"},{name:"Paul"},{name:"Jason"}];
    ```

    Ce code remplit le tableau `namesArray` avec plusieurs objets. Pendant cette affectation de variables, la liaison que vous avez établie précédemment entre les composants DataHolder et DataGrid s'exécute.
11. Testez le fichier en choisissant Contrôle > Tester l'animation.

Classe DataHolder

Héritage MovieClip > DataHolder

Nom de classe ActionScript mx.data.components.DataHolder

Le composant DataHolder est un référentiel pour les données et un moyen de générer des événements lors de la modification de ces données. Il permet essentiellement de stocker les données et joue le rôle de connecteur entre d'autres composants qui utilisent la liaison de données.

Initialement, le composant DataHolder possède une seule propriété de liaison, appelée `data`. L'onglet Schéma de l'inspecteur de composants vous permet de lui en ajouter d'autres.

Propriétés de la classe DataHolder

Le tableau suivant présente les propriétés de la classe DataHolder.

Propriété	Description
`DataHolder.data`	Propriété de liaison par défaut du composant DataHolder.

DataHolder.data

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`dataHolder``.data`

`Description`

Propriété ; l'élément par défaut dans un schéma d'objet DataHolder. Cette propriété n'est pas un membre « permanent » du composant DataHolder. Il s'agit plutôt de la propriété de liaison par défaut pour chaque occurrence du composant. Dans l'onglet Schéma de l'inspecteur de composants, vous pouvez ajouter vos propres propriétés de liaison ou supprimer la propriété `data` par défaut.

Pour plus d'informations sur l'utilisation de l'onglet Schéma, reportez-vous à *Utilisation des schémas dans l'onglet Schéma* du manuel *Utilisation de Flash*.

Exemple

Pour consulter un exemple détaillé d'utilisation de ce composant, reportez-vous à « Création d'une application avec le composant DataHolder », à la page 329.

Le code suivant présente un exemple simple qui alimente le composant DataHolder avec des données correspondant à une variable. Pour tester l'application, saisissez une valeur dans la zone de texte, puis cliquez sur l'occurrence `addDate_btn`. La valeur est alors ajoutée au composant DataHolder. Cliquez sur l'occurrence `dumpDataHolder_btn` pour suivre le contenu du composant DataHolder.

```
// Faites glisser deux composants Button sur la scène (addDate_btn et
// dumpDataHolder_btn), un composant TextInput (myDate_txt) et
// un composant DataHolder (myDataHolder).
// Ajoutez le code ActionScript suivant à l'image 1 :

var dhListener:Object = new Object();
dhListener.click = function() {
  trace("dumping DataHolder");
  trace("  " + myDataHolder.myDate);
  trace("");
};
var dateListener:Object = new Object();
dateListener.click = function() {
  myDataHolder.myDate = myDate_txt.text;
  trace("added value");
};
this.dumpDataHolder_btn.addEventListener("click", dhListener);
this.addDate_btn.addEventListener("click", dateListener);
```

CHAPITRE 12

API DataProvider

12

L'API DataProvider est un jeu de méthodes et de propriétés que doit posséder une source de données pour qu'une classe basée sur des listes puisse communiquer avec elle. Arrays, RecordSets et DataSet implémentent tous cette API. Vous pouvez créer une classe compatible DataProvider en implémentant toutes les méthodes et propriétés présentées dans cette section. Un composant basé sur des listes peut ensuite utiliser cette classe comme fournisseur de données.

REMARQUE

L'API DataProvider est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Classe DataProvider

Nom de classe ActionScript mx.controls.listclasses.DataProvider

Les méthodes de la classe DataProvider vous permettent d'interroger et de modifier les données de tout composant qui affiche des données (également appelé *affichage*). L'API DataProvider diffuse également des événements `change` lorsque les données sont modifiées. Plusieurs affichages peuvent utiliser le même fournisseur de données et recevoir les événements `change`.

Un fournisseur de données est une série linéaire (comme un tableau) d'éléments. Chaque élément est un objet composé de plusieurs champs de données. Vous pouvez accéder à ces éléments par leur index (comme dans un tableau), en utilisant `DataProvider.getItemAt()`.

Les fournisseurs de données sont généralement utilisés avec des tableaux. Les composants de données appliquent toutes les méthodes de l'API DataProvider à `Array.prototype` lorsqu'un objet Array se trouve dans la même image ou le même écran qu'un composant de données. Cela vous permet d'utiliser tout tableau existant en tant que données pour les affichages qui ont une propriété `dataProvider`.

Grâce à l'API DataProvider, les composants qui permettent l'affichage de données (DataGrid, List, Tree, etc.) peuvent également afficher des objets Flash Remoting RecordSet et des données du composant DataSet. L'API DataProvider est le langage qu'utilisent les composants de données pour communiquer avec leurs fournisseurs de données.

Dans la documentation d'Adobe Flash, « DataProvider » correspond au nom de la classe, `dataProvider` à une propriété de n'importe quel composant se comportant comme une vue pour les données, et « fournisseur de données », au terme générique utilisé pour désigner une source de données.

Méthodes de l'API DataProvider

Le tableau suivant présente les méthodes de l'API DataProvider.

Méthode	Description
`DataProvider.addItem()`	Ajoute un élément à la fin du fournisseur de données.
`DataProvider.addItemAt()`	Ajoute un élément au fournisseur de données à l'emplacement spécifié.
`DataProvider.editField()`	Modifie un champ du fournisseur de données.
`DataProvider.getEditingData()`	Obtient les données en vue d'une modification à partir d'un fournisseur de données.
`DataProvider.getItemAt()`	Obtient une référence à l'élément à l'emplacement spécifié.
`DataProvider.getItemID()`	Renvoie l'ID unique de l'élément.
`DataProvider.removeAll()`	Supprime tous les éléments du fournisseur de données.
`DataProvider.removeItemAt()`	Supprime un élément du fournisseur de données à l'emplacement spécifié.
`DataProvider.replaceItemAt()`	Remplace l'élément par un autre à l'emplacement spécifié.
`DataProvider.sortItems()`	Trie les éléments du fournisseur de données via la fonction de comparaison ou les options de tri spécifiées.
`DataProvider.sortItemsBy()`	Trie les éléments du fournisseur de données par ordre alphabétique ou numérique, dans l'ordre spécifié et selon le critère du nom de champ spécifié.

Propriétés de l'API DataProvider

Le tableau suivant présente les propriétés de l'API DataProvider.

Propriété	Description
`DataProvider.length`	Nombre d'éléments dans un fournisseur de données.

Evénements de l'API DataProvider

Le tableau suivant présente les événements de l'API DataProvider.

Evénement	Description
`DataProvider.modelChanged`	Diffusé lorsque le fournisseur de données est modifié.

DataProvider.addItem()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDP.addItem(*item*)

Paramètres

item Objet contenant des données. Constitue un élément de fournisseur de données.

Valeur renvoyée

Aucune.

Description

Méthode qui ajoute un nouvel élément à la fin du fournisseur de données. Cette méthode déclenche l'événement `modelChanged` avec le nom d'événement `addItems`.

Exemple

L'exemple suivant ajoute un élément à la fin du fournisseur de données `myDP` :

```
myDP.addItem({label : "this is an Item"});
```

DataProvider.addItemAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDP.addItemAt(*index*, *item*)

Paramètres

index Nombre supérieur ou égal à 0. Ce nombre définit la position à laquelle l'élément doit être inséré. Il s'agit de l'index du nouvel élément.

item Objet contenant les données de l'élément.

Valeur renvoyée

Aucune.

Description

Méthode qui ajoute un nouvel élément au fournisseur de données à l'index spécifié. Les index supérieurs à la longueur du fournisseur de données sont ignorés.

Cette méthode déclenche l'événement `modelChanged` avec le nom d'événement `addItems`.

Exemple

L'exemple suivant ajoute un élément au fournisseur de données `myDP` à la quatrième place :

```
myDP.addItemAt(3, {label : "this is the fourth Item"});
```

DataProvider.editField()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDP.editField(*index*, *fieldName*, *newData*)

Paramètres

index Nombre supérieur ou égal à 0. Index de l'élément.

fieldName Chaîne indiquant le nom du champ à modifier dans l'élément.

newData Nouvelles données à placer dans le fournisseur de données.

Valeur renvoyée

Aucune.

Description

Méthode qui modifie un champ dans le fournisseur de données.

Cette méthode déclenche l'événement `modelChanged` avec le nom d'événement `updateField`.

Exemple

Le code suivant modifie le champ `label` du troisième élément :

```
myDP.editField(2, "label", "mynewData");
```

DataProvider.getEditingData()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myDP`.getEditingData(`*index, fieldName*`)`

Paramètres

index Nombre supérieur ou égal à 0 et inférieur à `DataProvider.length`. Ce nombre correspond à l'index de l'élément à récupérer.

fieldName Chaîne indiquant le nom du champ modifié.

Valeur renvoyée

Les données formatées modifiables à utiliser.

Description

Méthode qui récupère les données pour leur modification à partir d'un fournisseur de données. Cela permet au modèle de données de proposer plusieurs formats de données pour modification et affichage.

Exemple

Le code suivant obtient une chaîne modifiable pour le champ « price » :

```
trace(myDP.getEditingData(4, "price");
```

DataProvider.getItemAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
myDP.getItemAt(index)
```

Paramètres

index Nombre supérieur ou égal à 0 et inférieur à `DataProvider.length`. Ce nombre correspond à l'index de l'élément à récupérer.

Valeur renvoyée

Une référence à l'élément récupéré ; « undefined » si l'index est hors limites.

Description

Méthode qui récupère une référence à l'élément à un emplacement spécifié.

Exemple

Le code suivant affiche l'étiquette du cinquième élément :

```
trace(myDP.getItemAt(4).label);
```

DataProvider.getItemID()

Disponibilité
Flash Player 6 (6.0.79.0).

Edition
Flash MX Professional 2004.

Utilisation
myDP.getItemID(*index*)

Paramètres
index Nombre supérieur ou égal à 0.

Valeur renvoyée
Un nombre correspondant à l'ID unique de l'élément.

Description
Méthode qui renvoie l'ID unique de l'élément. Cette méthode est principalement utilisée pour suivre la sélection. L'ID est utilisé dans les composants de données pour conserver la liste des éléments sélectionnés.

Exemple
Cet exemple obtient l'ID du quatrième élément :

```
var ID = myDP.getItemID(3);
```

DataProvider.length

Disponibilité
Flash Player 6 (6.0.79.0).

Edition
Flash MX Professional 2004.

Utilisation
myDP.length

Description
Propriété (lecture seule) : nombre d'éléments dans le fournisseur de données.

Exemple
Cet exemple envoie le nombre d'éléments du fournisseur de données `myArray` vers le panneau Sortie :

```
trace(myArray.length);
```

DataProvider.modelChanged

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
listenerObject = new Object();
listenerObject.modelChanged = function(eventObject){
  // Insérer votre code ici.
}
myMenu.addEventListener("modelChanged", listenerObject)
```

Description

Evénement : diffusé à tous ses écouteurs d'affichage lorsque le fournisseur de données est modifié. Pour ajouter un écouteur à un modèle, affectez sa propriété `dataProvider`.

Les composants utilisent un modèle dispatcher(diffuseur)/écouteur d'événement. Lorsqu'un fournisseur de données est modifié de quelque manière que ce soit, il diffuse un événement `modelChanged`, récupéré par les composants de données qui mettent à jour leurs affichages pour faire apparaître ces modifications.

L'objet événement de l'événement `Menu.modelChanged` compte cinq propriétés supplémentaires :

- `eventName` Propriété qui permet de classer en sous-catégories les événements `modelChanged`. Les composants de données utilisent cette information pour éviter d'actualiser l'occurrence de composant (affichage) qui utilise le fournisseur de données. La propriété `eventName` reconnaît les valeurs suivantes :
  - `updateAll` L'intégralité de la vue doit être actualisée, en excluant la position de défilement.
  - `addItems` Une série d'éléments a été ajoutée.
  - `removeItems` Une série d'éléments a été supprimée.
  - `updateItems` Une série d'éléments doit être actualisée.
  - `sort` Les données ont été triées.
  - `updateField` Un champ d'un élément doit être modifié et actualisé.
  - `updateColumn` Une définition de champ complète dans le fournisseur de données doit être actualisée.

    - `filterModel` Le modèle a été filtré et l'affichage doit être actualisé (réinitialisez la position de défilement).
    - `schemaLoaded` La définition de champ du fournisseur de données a été déclarée.
- `firstItem` Index du premier élément affecté.
- `lastItem` Index du dernier élément affecté. Si un seul élément est affecté, la valeur est égale à `firstItem`.
- `removedIDs` Tableau des identifiants d'éléments supprimés.
- `fieldName` Chaîne indiquant le nom du champ affecté.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `listener` est défini, puis transmis à la méthode `addEventListener()` comme second paramètre. L'objet événement est capturé par le gestionnaire `modelChanged` dans le paramètre `evt`. Lorsque l'événement `modelChanged` est diffusé, une instruction `trace` est envoyée au panneau Sortie.

```
listener = new Object();
listener.modelChanged = function(evt){
    trace(evt.eventName);
}
myList.addEventListener("modelChanged", listener);
```

DataProvider.removeAll()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
myDP.removeAll()
```

Paramètres

Aucun.

Valeur renvoyée

Aucune.

Description

Méthode qui supprime tous les éléments du fournisseur de données. Cette méthode déclenche l'événement `modelChanged` avec le nom d'événement `removeItems`.

Exemple

Cet exemple supprime tous les éléments du fournisseur de données :

```
myDP.removeAll();
```

DataProvider.removeItemAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`myDP``.removeItemAt(`*`index`*`)`

Paramètres

`index` Nombre supérieur ou égal à 0 correspondant à l'index de l'élément à supprimer.

Valeur renvoyée

Aucune.

Description

Méthode qui supprime l'élément à l'emplacement d'index spécifié. Un index disparaît parmi ceux qui sont situés après l'index supprimé.

Cette méthode déclenche l'événement `modelChanged` avec le nom d'événement `removeItems`.

Exemple

Cet exemple supprime l'élément situé en quatrième position :

```
myDP.removeItemAt(3);
```

DataProvider.replaceItemAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
myDP.replaceItemAt(index, item)
```

Paramètres

index Nombre supérieur ou égal à 0 correspondant à l'index de l'élément à modifier.

item Objet correspondant au nouvel élément.

Valeur renvoyée

Aucune.

Description

Méthode qui remplace le contenu de l'élément à l'emplacement d'index spécifié. Cette méthode déclenche l'événement `modelChanged` avec le nom d'événement `updateItems`.

Exemple

Cet exemple remplace l'élément à l'index 3 par l'élément possédant l'étiquette « new label » :

```
myDP.replaceItemAt(3, {label : "new label"});
```

DataProvider.sortItems()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
myDP.sortItems([compareFunc], [optionsFlag])
```

Paramètres

compareFunc Référence à une fonction permettant de comparer deux éléments pour déterminer leur ordre de tri. Pour plus d'informations, reportez-vous à la méthode `Array.sort()` dans le *Guide de référence du langage ActionScript 2.0*. Ce paramètre est facultatif.

optionsFlag Permet d'effectuer plusieurs types de tris dans un seul tableau sans avoir à répliquer l'intégralité du tableau ou à le trier à plusieurs reprises. Ce paramètre est facultatif.

Les valeurs possibles pour *optionsFlag* sont les suivantes :

- `Array.DESCENDING` trie par ordre décroissant.
- `Array.CASEINSENSITIVE` trie sans respecter la casse.
- `Array.NUMERIC` trie par ordre numérique si les deux éléments comparés sont des nombres. S'il ne s'agit pas de nombres, effectuez une comparaison de type chaîne (qui peut être non sensible à la casse si l'indicateur est spécifié).
- `Array.UNIQUESORT` renvoie un code d'erreur (0) au lieu d'un tableau trié si deux objets du tableau sont identiques ou comportent des champs de tri identiques.
- `Array.RETURNINDEXEDARRAY` renvoie un tableau d'index de nombres entiers correspondant au résultat du tri. Par exemple, le tableau suivant renverra la seconde ligne de code et ne sera pas modifié :

```
["a", "d", "c", "b"]
[0, 3, 2, 1]
```

Vous pouvez combiner ces options en une seule valeur. Par exemple, le code suivant associe les options 3 et 1 :

```
array.sort (Array.NUMERIC | Array.DESCENDING)
```

Valeur renvoyée

Aucune.

Description

Méthode qui trie les éléments du fournisseur de données selon la fonction de comparaison ou les options de tri spécifiées.

Cette méthode déclenche l'événement `modelChanged` avec le nom d'événement `sort`.

Exemple

Cet exemple trie selon des étiquettes en majuscules. Les éléments `a` et `b` sont transmis à la fonction et possèdent les champs `label` et `data` :

```
myList.sortItems(upperCaseFunc);
function upperCaseFunc(a,b){
    return a.label.toUpperCase() > b.label.toUpperCase();
}
```

DataProvider.sortItemsBy()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
myDP.sortItemsBy(fieldName, optionsFlag)
myDP.sortItemsBy(fieldName, order)
```

Paramètres

fieldName Chaîne spécifiant le nom du champ à utiliser pour le tri. Cette valeur est en général « `label` » ou « `data` ».

order Chaîne spécifiant si le tri des éléments doit être effectué par ordre croissant (« `ASC` ») ou décroissant (« `DESC` »).

optionsFlag Permet d'effectuer plusieurs types de tris dans un seul tableau sans avoir à répliquer l'intégralité du tableau ou à le trier à plusieurs reprises. Ce paramètre est facultatif.

Les valeurs possibles pour *optionsFlag* sont les suivantes :

- `Array.DESCENDING` trie par ordre décroissant.
- `Array.CASEINSENSITIVE` trie sans respecter la casse.
- `Array.NUMERIC` trie par ordre numérique si les deux éléments comparés sont des nombres. S'il ne s'agit pas de nombres, effectuez une comparaison de type chaîne (qui peut être non sensible à la casse si l'indicateur est spécifié).
- `Array.UNIQUESORT` si deux objets du tableau sont identiques ou comportent des champs de tri identiques, cette méthode renvoie un code d'erreur (0) au lieu d'un tableau trié.
- `Array.RETURNINDEXEDARRAY` renvoie un tableau d'index de nombres entiers correspondant au résultat du tri. Par exemple, le tableau suivant renverra la seconde ligne de code et ne sera pas modifié :

```
["a", "d", "c", "b"]
[0, 3, 2, 1]
```

Vous pouvez combiner ces options en une seule valeur. Par exemple, le code suivant associe les options 3 et 1 :

```
array.sort (Array.NUMERIC | Array.DESCENDING)
```

Valeur renvoyée

Aucune.

Description

Méthode qui trie les éléments du fournisseur de données dans l'ordre spécifié, via le nom de champ spécifié. Si les éléments *`fieldName`* sont une combinaison de chaînes de texte et de nombres entiers, les éléments entiers sont indiqués en premier. Le paramètre *`fieldName`* est généralement « `label` » ou « `data` », mais les programmeurs expérimentés peuvent spécifier n'importe quelle primitive.

Cette méthode déclenche l'événement `modelChanged` avec le nom d'événement `sort`.

Elle représente le moyen le plus rapide pour trier les données dans un composant. Elle permet également de conserver l'état de sélection du composant. La méthode `sortItemsBy()` est rapide, car elle n'exécute pas de code ActionScript pendant le tri. La méthode `sortItems()` doit exécuter une fonction de comparaison ActionScript et se révèle donc plus lente.

Exemple

Le code suivant trie les éléments d'une liste par ordre croissant en utilisant leurs étiquettes :

```
myDP.sortItemsBy("label", "ASC");
```

CHAPITRE 13

Composant DateChooser

13

Ce composant est un calendrier qui permet aux utilisateurs de sélectionner une date. Ses boutons leur permettent de faire défiler les mois et de cliquer sur la date de leur choix. Vous pouvez définir des paramètres qui indiquent les noms des mois et des jours, le premier jour de la semaine et les dates désactivées, et mettent la date du jour en surbrillance.

REMARQUE	Le composant DateChooser est pris en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Un aperçu en direct de chaque occurrence DateChooser reflète les valeurs indiquées par l'inspecteur Propriétés ou des composants pendant la programmation.

Utilisation du composant DateChooser

Le composant DateChooser peut être exploité pour chaque sélection de date par l'utilisateur. Par exemple, vous pouvez l'utiliser dans un système de réservation de chambres d'hôtel comprenant des dates sélectionnables et des dates désactivées. Vous pouvez également l'utiliser dans une application qui affiche les événements en cours, spectacles ou réunions, lorsque l'utilisateur sélectionne une date.

Paramètres du composant DateChooser

Dans l'inspecteur Propriétés ou l'inspecteur de composants (option de menu Fenêtre > Inspecteur de composants), vous pouvez définir les paramètres de création suivants pour chaque occurrence DateChooser :

dayNames définit les noms des jours de la semaine. La valeur est un tableau et la valeur par défaut est `["S", "M", "T", "W", "T", "F", "S"]`.

disabledDays indique les jours désactivés de la semaine. Ce paramètre est un tableau qui peut comprendre un maximum de sept valeurs. La valeur par défaut est `[]` (tableau vide).

firstDayOfWeek indique le jour de la semaine (0-6, 0 étant le premier élément du tableau `dayNames`) affiché dans la première colonne du sélecteur de dates. Cette propriété modifie l'ordre d'affichage des colonnes des jours.

monthNames définit les noms des mois affichés dans la ligne d'en-tête du calendrier. La valeur est un tableau et la valeur par défaut est `["January", "February", "March", "April", "May", "June", "July", "August", "September", "October","November", "December"]`.

showToday indique si la date du jour doit être mise en surbrillance. La valeur par défaut est `true`.

Dans l'inspecteur des composants, vous pouvez définir les paramètres supplémentaires suivants pour chaque occurrence du composant DateChooser (Fenêtre > Inspecteur de composants) :

enabled est une valeur booléenne qui indique si le composant peut recevoir le focus et l'entrée. La valeur par défaut est `true`.

visible est une valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`). La valeur par défaut est `true`.

REMARQUE	Les propriétés minHeight et minWidth sont utilisées par des routines de dimensionnement internes. Elles sont définies dans UIObject et remplacées par différents composants selon vos besoins. Ces propriétés peuvent être utilisées pour créer un gestionnaire de dispositions personnalisées. Dans le cas contraire, la définition de ces propriétés dans l'inspecteur de composants n'a aucun effet visible.

Vous pouvez rédiger du code ActionScript pour contrôler ces options et d'autres options du composant DateChooser à l'aide de ses propriétés, méthodes et événements. Pour plus d'informations, voir « Classe DateChooser », à la page 354.

Création d'une application avec le composant DateChooser

La procédure suivante décrit l'ajout d'un composant DateChooser à une application lors de la programmation. Dans cet exemple, le sélecteur de dates permet à l'utilisateur de choisir une date dans un système de réservation de vols. Toutes les dates précédant le 15 octobre doivent être désactivées, de même que les lundis et une plage de dates du mois de décembre afin de créer une période de vacances.

Pour créer une application avec le composant DateChooser :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Dans le panneau Composants, double-cliquez sur le composant DateChooser afin de l'ajouter sur la scène.
3. Dans l'inspecteur Propriétés, entrez le nom d'occurrence **flightCalendar**.
4. Dans le panneau Actions, entrez le code suivant dans l'image 1 du scénario pour définir la plage des dates sélectionnables :

   ```
   flightCalendar.selectableRange = {rangeStart:new Date(2003, 9, 15),
     rangeEnd:new Date(2003, 11, 31)}
   ```

   Ce code affecte une valeur à la propriété `selectableRange` dans un objet ActionScript qui contient deux objets Date dont les variables sont nommées `rangeStart` et `rangeEnd`. Il définit les limites inférieure et supérieure de la plage dans laquelle l'utilisateur peut choisir une date.
5. Dans le panneau Actions, entrez le code suivant dans l'image 1 du scénario pour définir la plage de dates désactivées correspondant aux vacances :

   ```
   flightCalendar.disabledRanges = [{rangeStart: new Date(2003, 11, 15),
     rangeEnd: new Date(2003, 11, 26)}];
   ```

6. Dans le panneau Actions, entrez le code suivant dans l'image 1 du scénario pour désactiver les lundis :

   ```
   flightCalendar.disabledDays=[1];
   ```

7. Choisissez Contrôle > Tester l'animation.

Pour créer une occurrence du composant DateChooser à l'aide d'ActionScript :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser le composant DateChooser du panneau Composants jusqu'à la bibliothèque du document actif.
3. Sélectionnez la première image dans le scénario principal, ouvrez le panneau Actions et indiquez le code suivant :

   ```
   this.createClassObject(mx.controls.DateChooser, "my_dc", 1);
   ```

   Ce script utilise la méthode « UIObject.createClassObject() », à la page 1416 pour créer l'occurrence du composant DateChooser, puis dimensionne et positionne la grille.
4. Sélectionnez Contrôle > Tester l'animation.

Personnalisation du composant DateChooser

Vous pouvez orienter un composant DateChooser horizontalement et verticalement pendant la création et l'exécution. Lors de la programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez la méthode `setSize()` (voir `UIObject.setSize()`).

Utilisation de styles avec le composant DateChooser

Vous pouvez définir des propriétés de style pour modifier l'aspect de l'occurrence DateChooser. Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, consultez « Utilisation de styles pour personnaliser la couleur et le texte des composants » dans *Utilisation des composants ActionScript 2.0*.

Un composant DateChooser prend en charge les styles suivants :

Style	Thème	Description
`themeColor`	Halo	Couleur de l'éclat pour les dates survolées et sélectionnées. `"haloGreen"`, `"haloBlue"` et `"haloOrange"` sont des valeurs possibles. La valeur par défaut est `"haloGreen"`.
`backgroundColor`	Les deux	Couleur d'arrière-plan. La valeur par défaut est 0xEFEBEF (gris clair).
`borderColor`	Les deux	Couleur de bordure. La valeur par défaut est 0x919999. Le composant DateChooser utilise comme bordure une ligne unie et large d'un pixel. Cette bordure n'est pas modifiable par l'application d'enveloppes ou de styles.
`headerColor`	Les deux	Couleur d'arrière-plan de l'en-tête du composant. La couleur par défaut est le blanc.
`rollOverColor`	Les deux	Couleur d'arrière-plan d'une date survolée. La couleur par défaut est 0xE3FFD6 (vert vif) pour le thème Halo et 0xAAAAAA (gris clair) pour le thème Sample.
`selectionColor`	Les deux	Couleur d'arrière-plan de la date sélectionnée. La couleur par défaut est 0xCDFFC1 (vert clair) pour le thème Halo et 0xEEEEEE (gris très clair) pour le thème Sample.

Style	Thème	Description
todayColor	Les deux	Couleur d'arrière-plan de la date du jour. La valeur par défaut est 0x666666 (gris foncé).
color	Les deux	Couleur du texte. La valeur par défaut est 0x0B333C pour le thème Halo et vide pour le thème Sample.
disabledColor	Les deux	Couleur du texte lorsque le composant est désactivé. La couleur par défaut est 0x848384 (gris foncé).
embedFonts	Les deux	Valeur booléenne qui indique si la police spécifiée dans fontFamily est de type intégré. Ce style doit être défini sur true si fontFamily fait référence à une police intégrée. Dans le cas contraire, la police intégrée n'est pas utilisée. Si ce style est défini sur true et que fontFamily ne fait pas référence à une police intégrée, aucun texte n'est affiché. La valeur par défaut est false.
fontFamily	Les deux	Nom de la police du texte. La valeur par défaut est "_sans".
fontSize	Les deux	Taille de la police, en points. La valeur par défaut est 10.
fontStyle	Les deux	Style de police : "normal" ou "italic". La valeur par défaut est "normal".
fontWeight	Les deux	Epaisseur de la police : "none" ou "bold". La valeur par défaut est "none". Tous les composants peuvent également accepter la valeur « normal » au lieu de « none » pendant un appel de setStyle(), mais les prochains appels de getStyle() renvoient « none ».
textDecoration	Les deux	Décoration du texte : "none" ou "underline". La valeur par défaut est "none".

Le composant DateChooser utilise quatre catégories de texte pour afficher le nom du mois, les jours de la semaine, la date du jour et les dates ordinaires. Les propriétés de style de texte définies dans le composant DateChooser lui-même contrôlent le texte des dates ordinaires et fournissent les valeurs par défaut de l'autre texte. Pour définir des styles de texte pour des catégories de texte spécifiques, servez-vous des déclarations de styles suivantes au niveau de la classe.

Nom de la déclaration	Description
HeaderDateText	Nom du mois.
WeekDayStyle	Jours de la semaine.
TodayStyle	Date du jour.

L'exemple suivant montre comment définir le nom du mois et les jours de la semaine en rouge foncé.

```
_global.styles.HeaderDateText.setStyle("color", 0x660000);
_global.styles.WeekDayStyle.setStyle("color", 0x660000);
```

Utilisation des enveloppes avec le composant DateChooser

Pour représenter les boutons Précédent et Suivant du mois et l'indicateur du jour, le composant DateChooser utilise des enveloppes. Pour appliquer des enveloppes au composant DateChooser lors de la création, modifiez les symboles d'enveloppes dans le dossier Flash UI Components 2/Themes/MMDefault/DateChooser Assets/States dans la bibliothèque de l'un des fichiers FLA de thèmes. Pour plus d'informations, reportez-vous à « A propos de l'application d'enveloppes aux composants » dans *Utilisation des composants ActionScript 2.0*.

Seuls les boutons de défilement des mois peuvent être enveloppés dynamiquement dans ce composant. Un composant DateChooser utilise les propriétés d'enveloppe suivantes :

Propriété	Description
`backMonthButtonUpSymbolName`	Etat relevé du bouton Précédent du mois. La valeur par défaut est `backMonthUp`.
`backMonthButtonDownSymbolName`	Etat enfoncé du bouton Précédent du mois. La valeur par défaut est `backMonthDown`.
`backMonthButtonDisabledSymbolName`	Etat désactivé du bouton Précédent du mois. La valeur par défaut est `backMonthDisabled`.
`fwdMonthButtonUpSymbolName`	Etat relevé du bouton Suivant du mois. La valeur par défaut est `fwdMonthUp`.
`fwdMonthButtonDownSymbolName`	Etat enfoncé du bouton Suivant du mois. La valeur par défaut est `fwdMonthDown`.
`fwdMonthButtonDisabledSymbolName`	Etat désactivé du bouton Suivant du mois. La valeur par défaut est `fwdMonthDisabled`.

Les symboles de bouton sont utilisés en l'état, sans application de couleurs ni de redimensionnement. La taille est déterminée par le symbole lors de la programmation.

Pour créer des symboles de clip pour les enveloppes DateChooser :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Choisissez Fichier > Importer > Ouvrir une bibliothèque externe et sélectionnez le fichier HaloTheme.fla.

   Ce fichier est stocké dans le dossier Configuration au niveau de l'application. Pour connaître son emplacement exact dans votre système d'exploitation, reportez-vous à « Présentation des thèmes » dans *Utilisation des composants ActionScript 2.0*.
3. Dans le panneau Bibliothèque du thème, choisissez Flash UI Components 2/Themes/MMDefault et faites glisser le dossier DateChooser Assets dans la bibliothèque de votre document.
4. Développez le dossier DateChooser Assets/States dans la bibliothèque de votre document.
5. Ouvrez les symboles que vous souhaitez personnaliser pour la modification.

   Par exemple, ouvrez le symbole backMonthDown.
6. Personnalisez le symbole selon vos besoins.

   Par exemple, modifiez la teinte de la flèche en rouge.
7. Répétez les étapes 5 et 6 pour tous les symboles devant être personnalisés.

   Par exemple, définissez la teinte du symbole de la flèche bas Suivant sur celle de la flèche Précédent.
8. Cliquez sur le bouton Précédent pour revenir au scénario principal.
9. Faites glisser un composant DateChooser vers la scène.
10. Sélectionnez Contrôle > Tester l'animation.

REMARQUE

Le dossier DateChooser Assets/States contient également un dossier Day Skins comportant un unique élément d'enveloppe, `cal_todayIndicator`. Cet élément peut être modifié pendant la programmation pour personnaliser l'indicateur du jour. Il ne peut cependant pas être modifié dynamiquement dans une occurrence DateChooser particulière pour utiliser un autre symbole. En outre, le symbole `cal_todayIndicator` doit être un graphique de couleur unie car le composant DateChooser applique la couleur `todayColor` à l'ensemble du graphique. Ce dernier peut présenter des découpes, mais n'oubliez pas que dans ce cas la couleur par défaut du texte de la date du jour et de l'arrière-plan du composant DateChooser est le blanc et, de ce fait, qu'une découpe centrale de l'élément d'enveloppe de l'indicateur rendrait la date du jour illisible, sauf si sa couleur d'arrière-plan ou son texte sont également modifiés.

Classe DateChooser

Héritage MovieClip > Classe UIObject > Classe UIComponent > DateChooser

Nom de classe ActionScript mx.controls.DateChooser

Les propriétés de la classe DateChooser vous permettent d'accéder à la date sélectionnée et au mois et à l'année affichés. Vous pouvez également définir les noms des jours et des mois, indiquer les dates désactivées et les dates sélectionnables, définir le premier jour de la semaine et indiquer si la date du jour doit être mise en surbrillance.

La définition d'une propriété de la classe DateChooser avec ActionScript annule le paramètre du même nom défini dans l'inspecteur Propriétés ou des composants.

Toutes les classes de composants ont une propriété `version` qui correspond à une propriété de classe. Les propriétés de classe ne sont disponibles que dans la classe elle-même. La propriété `version` renvoie une chaîne qui indique la version du composant. Pour accéder à cette propriété, utilisez le code suivant :

```
trace(mx.controls.DateChooser.version);
```

REMARQUE

Le code `trace(myDC.version);` renvoie `undefined`.

Méthodes de la classe DateChooser

La classe DateChooser ne présente pas de méthodes exclusives.

Méthodes héritées de la classe UIObject

Le tableau suivant énumère les méthodes de la classe DateChooser héritées de la classe UIObject. Pour appeler ces méthodes à partir de l'objet DateChooser, utilisez le formulaire *`dateChooserInstance.methodName`*.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.

Méthode	Description
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
`UIObject.setSize()`	Redimensionne l'objet à la taille demandée.
`UIObject.setSkin()`	Définit une enveloppe dans l'objet.
`UIObject.setStyle()`	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant énumère les méthodes de la classe DateChooser héritées de la classe UIComponent. Pour appeler ces méthodes à partir de l'objet DateChooser, utilisez le formulaire *`dateChooserInstance.methodName`*.

Méthode	Description
`UIComponent.getFocus()`	Renvoie une référence à l'objet ayant le focus.
`UIComponent.setFocus()`	Attribue le focus à l'occurrence de composant.

Propriétés de la classe DateChooser

Le tableau suivant présente les propriétés réservées à la classe DateChooser.

Propriété	Description
`DateChooser.dayNames`	Tableau indiquant les noms des jours de la semaine.
`DateChooser.disabledDays`	Tableau indiquant les jours de la semaine qui sont désactivés pour toutes les dates applicables dans le sélecteur.
`DateChooser.disabledRanges`	Plage de dates (ou date unique) désactivées.
`DateChooser.displayedMonth`	Nombre indiquant un élément du tableau `monthNames` à afficher dans le sélecteur de dates.
`DateChooser.displayedYear`	Nombre indiquant l'année à afficher.
`DateChooser.firstDayOfWeek`	Nombre indiquant un élément du tableau `dayNames` à afficher dans la première colonne du sélecteur de dates.

Propriété	Description
`DateChooser.monthNames`	Tableau de chaînes indiquant les noms des mois.
`DateChooser.selectableRange`	Plage de dates (ou date unique) sélectionnables.
`DateChooser.selectedDate`	Objet Date indiquant la date sélectionnée.
`DateChooser.showToday`	Valeur booléenne indiquant si la date du jour est mise en surbrillance.

Propriétés héritées de la classe UIObject

Le tableau suivant énumère les propriétés de la classe DateChooser héritées de la classe UIObject. Lors de l'accès à ces propriétés à partir de l'objet DateChooser, utilisez le formulaire *`dateChooserInstance.propertyName`*.

Propriété	Description
`UIObject.bottom`	Position du bord inférieur de l'objet par rapport au bord inférieur de son parent. Lecture seule.
`UIObject.height`	Hauteur de l'objet, en pixels. Lecture seule.
`UIObject.left`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.right`	Position du bord droit de l'objet par rapport au bord droit de son parent. Lecture seule.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Position du bord supérieur de l'objet par rapport à son parent. Lecture seule.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Largeur de l'objet, en pixels. Lecture seule.
`UIObject.x`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.y`	Bord supérieur de l'objet, en pixels. Lecture seule.

Propriétés héritées de la classe UIComponent

Le tableau suivant énumère les propriétés de la classe DateChooser héritées de la classe UIComponent. Lors de l'accès à ces propriétés à partir de l'objet DateChooser, utilisez le formulaire *dateChooserInstance.propertyName*.

Propriété	Description
UIComponent.enabled	Indique si le composant peut recevoir le focus et la saisie.
UIComponent.tabIndex	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Evénements de la classe DateChooser

Le tableau suivant présente les événements réservés à la classe DateChooser.

Evénement	Description
DateChooser.change	Diffusé lorsqu'une date est sélectionnée.
DateChooser.scroll	Diffusé lorsque l'utilisateur clique sur les boutons des mois.

Evénements hérités de la classe UIObject

Le tableau suivant énumère les événements de la classe DateChooser hérités de la classe UIObject.

Evénement	Description
UIObject.draw	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
UIObject.hide	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
UIObject.load	Diffusé lorsque des sous-objets sont créés.
UIObject.move	Diffusé lorsque l'objet a été déplacé.
UIObject.resize	Diffusé lorsqu'un objet a été redimensionné.
UIObject.reveal	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
UIObject.unload	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant énumère les événements de la classe DateChooser hérités de la classe UIComponent.

Evénement	Description
UIComponent.focusIn	Diffusé lorsqu'un objet reçoit le focus.
UIComponent.focusOut	Diffusé lorsqu'un objet perd le focus.
UIComponent.keyDown	Diffusé lorsqu'une touche est enfoncée.
UIComponent.keyUp	Diffusé lorsqu'une touche est relâchée.

DateChooser.change

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.change = function(eventObject:Object) {
  // ...
};
dateChooserInstance.addEventListener("change", listenerObject);
```

Utilisation 2 :

```
on (change) {
  //...
}
```

Description

Evénement diffusé à tous les écouteurs enregistrés lorsqu'une date est sélectionnée.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence du composant (*dataChooserInstance*) distribue un événement (ici, change) qui est géré par une fonction, également appelée *gestionnaire*, dans un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lors de son déclenchement, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. Chaque objet événement est doté de propriétés contenant des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le premier exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence DateChooser. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence du composant. Par exemple, le code suivant, associé au sélecteur de dates `my_dc`, envoie « _level0.my_dc » au panneau Sortie :

```
on (change) {
  trace(this);
}
```

Exemple

Cet exemple, écrit dans une image du scénario, envoie un message au panneau Sortie lorsqu'une occurrence DateChooser nommée `my_dc` est modifiée. La première ligne de code crée un objet écouteur intitulé `form`. La deuxième ligne définit une fonction pour l'événement `change` sur l'objet écouteur. La fonction comporte une instruction `trace()` qui utilise l'objet événement automatiquement transmis à cette fonction, ici `eventObj`, pour générer un message. La propriété `target` d'un objet événement est le composant qui a généré l'événement (dans cet exemple, `my_dc`).

```
// Création d'un objet écouteur.
var dcListener:Object = new Object();
dcListener.change = function(evt_obj:Object) {
 var thisDate:Date = evt_obj.target.selectedDate;
    trace("date selected: " +  thisDate);
};

// Ajout d'un objet écouteur au sélecteur de dates.
my_dc.addEventListener("change", dcListener);
```

DateChooser.dayNames

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dateChooserInstance.dayNames

Description

Propriété, tableau contenant les noms des jours de la semaine. Dimanche correspond au premier jour (à l'emplacement d'index 0) et le reste des jours suit, dans l'ordre. La valeur par défaut est `["S", "M", "T", "W", "T", "F", "S"]`.

Exemple

L'exemple suivant modifie la valeur des jours de la semaine :

```
my_dc.dayNames = new Array("Su", "Mo", "Tu", "We", "Th", "Fr", "Sa");
```

DateChooser.disabledDays

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dateChooserInstance.disabledDays

Description

Propriété, tableau indiquant les jours désactivés de la semaine. Toutes les dates du mois qui correspondent au jour spécifié sont désactivées. Les éléments de ce tableau peuvent avoir des valeurs comprises entre 0 (dimanche) et 6 (samedi). La valeur par défaut est `[]` (tableau vide).

Exemple

L'exemple suivant désactive les dimanches et samedis pour que les utilisateurs ne puissent choisir que des jours de semaine :

```
my_dc.disabledDays = [0, 6];
```

DateChooser.disabledRanges

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dateChooserInstance.disabledRanges

Description

Propriété qui désactive un jour ou une plage de jours. Cette propriété est un tableau d'objets. Chaque objet du tableau doit correspondre à un objet Date spécifiant le jour à désactiver ou à un objet contenant une des propriétés `rangeStart` ou `rangeEnd`, voire les deux, dont les valeurs doivent être un objet Date. Les propriétés `rangeStart` et `rangeEnd` décrivent les limites de la plage de dates. Si l'une ou l'autre des propriétés est omise, l'étendue est illimitée dans le sens correspondant.

La valeur par défaut de `disabledRanges` est `undefined`.

Lorsque vous définissez des dates pour la propriété `disabledRanges`, spécifiez des dates complètes. Par exemple, spécifiez `new Date(2003,6,24)` plutôt que `new Date()`. Si vous ne spécifiez pas une date complète, l'objet Date renvoie la date et l'heure du jour. Si vous ne spécifiez pas une heure, l'heure renvoie `00:00:00`.

Exemple

L'exemple suivant définit un tableau comprenant des objets Date `rangeStart` et `rangeEnd` qui désactivent les dates comprises entre le 7 mai et le 7 juin :

```
my_dc.disabledRanges = [{rangeStart: new Date(2003, 4, 7), rangeEnd: new
  Date(2003, 5, 7)}];
```

L'exemple suivant désactive toutes les dates ultérieures au 7 novembre :

```
my_dc.disabledRanges = [ {rangeStart: new Date(2003, 10, 7)} ];
```

L'exemple suivant désactive toutes les dates antérieures au 7 octobre :

```
my_dc.disabledRanges = [ {rangeEnd: new Date(2002, 9, 7)} ];
```

L'exemple suivant désactive uniquement le 7 décembre :

```
my_dc.disabledRanges = [ new Date(2003, 11, 7) ];
```

L'exemple suivant désactive le 7 avril et le 21 avril :

```
my_dc.disabledRanges = [ new Date(2003, 3, 7), new Date(2003, 3, 21)  ];
```

DateChooser.displayedMonth

Disponibilité
Flash Player 6 (6.0.79.0).

Edition
Flash MX Professional 2004.

Utilisation
dateChooserInstance.displayedMonth

Description
Propriété, nombre indiquant le mois à afficher. Le nombre indique un élément du tableau `monthNames`, 0 correspondant au premier mois. La valeur par défaut est le mois de la date du jour.

Exemple
L'exemple suivant définit le mois affiché sur Décembre :

```
my_dc.displayedMonth = 11;
```

Voir aussi
`DateChooser.displayedYear`

DateChooser.displayedYear

Disponibilité
Flash Player 6 (6.0.79.0).

Edition
Flash MX Professional 2004.

Utilisation
dateChooserInstance.displayedYear

Description
Propriété, nombre à quatre chiffres indiquant l'année affichée. La valeur par défaut est l'année en cours.

Exemple
L'exemple suivant définit l'année affichée sur 2010 :

```
my_dc.displayedYear = 2010;
```

Voir aussi
`DateChooser.displayedMonth`

DateChooser.firstDayOfWeek

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dateChooserInstance.firstDayOfWeek

Description

Propriété : chiffre indiquant le jour de la semaine (0-6, 0 correspondant au premier élément du tableau `dayNames`) affiché dans la première colonne du composant DateChooser.
La modification de cette propriété change l'ordre des colonnes des jours, mais n'a aucune incidence sur l'ordre de la propriété `dayNames`. La valeur par défaut est 0 (dimanche).

Exemple

L'exemple suivant définit le premier jour de la semaine sur Lundi :

```
// Définition du premier jour de la semaine sur lundi dans le calendrier.
my_dc.firstDayOfWeek = 1;

// Désactivation du jour 0 (dimanche). Même si lundi est à présent
// le premier jour dans le composant DateChooser, dimanche
// est toujours l'index 0 du tableau.
my_dc.disabledDays = [0];
```

Voir aussi

DateChooser.dayNames

DateChooser.monthNames

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
dateChooserInstance.monthNames
```

Description

Propriété, tableau de chaînes indiquant les noms des mois dans la partie supérieure du composant DateChooser. La valeur par défaut est `["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"]`.

Exemple

L'exemple suivant définit les noms de mois pour l'occurrence `my_dc` :

```
my_dc.monthNames = ["Jan", "Feb","Mar","Apr", "May", "June","July", "Aug",
  "Sept","Oct", "Nov", "Dec"];
```

DateChooser.scroll

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.scroll = function(eventObject:Object) {
  //...
}
dateChooserInstance.addEventListener("scroll", listenerObject)
```

Utilisation 2 :

```
on (scroll) {
  //...
}
```

Description

Evénement, diffusé à tous les écouteurs enregistrés lorsque l'utilisateur clique sur le bouton d'un mois.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence du composant (*myDC*) distribue un événement (ici, `scroll`) géré par une fonction (également appelée *gestionnaire*) dans un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée au déclenchement de l'événement. Lorsqu'il est déclenché, l'événement transmet automatiquement un objet événement (*eventObject*) à la méthode de l'objet écouteur. Chaque objet événement est doté de propriétés contenant des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. L'objet événement de l'événement scroll comprend une propriété supplémentaire, `detail`, qui peut prendre l'une des valeurs suivantes : `nextMonth`, `previousMonth`, `nextYear`, `previousYear`.

Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le premier exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence DateChooser. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence du composant. Par exemple, le code suivant, associé au sélecteur de dates `myDC`, envoie « _level0.myDC » au panneau Sortie :

```
on (scroll) {
  trace(this);
}
```

Exemple

Cet exemple, écrit dans une image du scénario, envoie un message au panneau Sortie lorsque l'utilisateur clique sur le bouton correspondant à un mois dans une occurrence DateChooser appelée `my_dc`. La première ligne de code crée un objet écouteur intitulé `form`. La deuxième ligne définit une fonction pour l'événement `scroll` sur l'objet écouteur. La fonction comporte une instruction `trace()` qui utilise l'objet événement automatiquement transmis à cette fonction (ici `evt_obj`) pour générer un message.

```
// Création d'un objet écouteur.
var dcListener:Object = new Object();
dcListener.scroll = function(evt_obj:Object) {
 trace(evt_obj.detail);
};

// Ajout d'un objet écouteur au sélecteur de dates.
my_dc.addEventListener("scroll", dcListener);
```

DateChooser.selectableRange

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dateChooserInstance.selectableRange

Description

Propriété qui définit une seule date sélectionnable ou une plage de dates sélectionnables. L'utilisateur ne peut pas faire défiler l'affichage au-delà de la plage sélectionnée. La valeur de cette propriété est un objet qui comprend deux objets Date appelés `rangeStart` et `rangeEnd`. Les propriétés `rangeStart` et `rangeEnd` désignent les limites de la plage de dates sélectionnables. Si seule la propriété `rangeStart` est définie, toutes les dates ultérieures à `rangeStart` sont activées. Si seule la propriété `rangeEnd` est définie, toutes les dates antérieures à `rangeEnd` sont activées. La valeur par défaut est `undefined`.

Si vous ne souhaitez activer qu'un seul jour, vous pouvez utiliser un objet Date unique comme valeur de la propriété `selectableRange`.

Lorsque vous définissez une date, indiquez la date complète. Par exemple, `new Date(2003,6,24)` plutôt que `new Date()`. Si vous ne spécifiez pas une date complète, l'objet Date renvoie la date et l'heure du jour. Si vous ne spécifiez pas une heure, l'heure renvoie `00:00:00`.

La valeur de `DateChooser.selectedDate` est définie sur `undefined` si elle se situe en dehors de la plage sélectionnable.

Les valeurs de `DateChooser.displayedMonth` et de `DateChooser.displayedYear` sont définies sur le dernier mois le plus proche dans la plage sélectionnable si le mois en cours n'est pas compris dans cette dernière. Par exemple, si le mois en cours affiché est août et que la plage sélectionnable va de juin à juillet 2003, le mois de juillet 2003 s'affiche.

Exemple

L'exemple suivant définit la plage sélectionnable entre le 7 mai (inclus) et le 7 juin (inclus) :

```
my_dc.selectableRange = {rangeStart: new Date(2001, 4, 7), rangeEnd: new
  Date(2003, 5, 7)};
```

L'exemple suivant définit la plage sélectionnable sur les dates ultérieures au 7 mai (inclus) :

```
my_dc.selectableRange = {rangeStart: new Date(2005, 4, 7)};
```

L'exemple suivant définit la plage sélectionnable sur les dates antérieures au 7 juin (inclus) :

```
my_dc.selectableRange = {rangeEnd: new Date(2005, 5, 7)};
```

L'exemple suivant définit la date sélectionnable uniquement au 7 juin :

```
my_dc.selectableRange = new Date(2005, 5, 7);
```

DateChooser.selectedDate

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dateChooserInstance.selectedDate

Description

Propriété : objet Date qui indique la date sélectionnée si la valeur est comprise dans la plage de valeurs de la propriété `selectableRange`. La valeur par défaut est `undefined`.

Vous ne pouvez pas définir la propriété `selectedDate` au sein d'une plage désactivée, hors d'une plage sélectionnable ou sur un jour désactivé. Si cette propriété est définie sur l'une de ces dates, la valeur est `undefined`.

Exemple

L'exemple suivant définit la date sélectionnée sur le 7 juin :

```
my_dc.selectedDate = new Date(2005, 5, 7);
```

DateChooser.showToday

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dateChooserInstance.showToday

Description

Propriété, valeur booléenne qui indique si la date du jour est mise en surbrillance. La valeur par défaut est `true`.

Exemple

L'exemple suivant désactive la mise en surbrillance de la date du jour :

```
my_dc.showToday = false;
```

CHAPITRE 14

Composant DataSet

14

Le composant DataSet vous permet d'exploiter des données sous forme de séries d'objets pouvant être indexés, triés, recherchés, filtrés et modifiés.

REMARQUE

Le composant DataSet est pris en charge uniquement si vous travaillez dans un document spécifiant Flash Player 7, ou une version ultérieure, et ActionScript 2.0 dans ses paramètres de publication.

La fonctionnalité du composant DataSet comprend DataSetIterator, jeu de méthodes pour parcourir et manipuler une collection de données, et DeltaPacket, jeu d'interfaces et de classes pour utiliser des mises à jour d'une collection de données. Dans la plupart des cas, vous n'utilisez pas directement ces classes et interfaces. Vous les utilisez indirectement par l'intermédiaire de méthodes fournies par la classe DataSet.

Les éléments gérés par le composant DataSet sont également appelés *objets de transfert*. Un objet de transfert expose les données métier qui résident sur le serveur avec des attributs publics ou des méthodes d'accesseur pour lire et écrire des données. Le composant DataSet permet aux développeurs d'utiliser des objets complexes côté client qui reflètent à l'identique les objets côté serveur ou, plus simplement, d'utiliser une collection d'objets anonymes, avec des attributs publics représentant les champs d'un enregistrement de données. Pour plus d'informations sur les objets de transfert, consultez la page Core J2EE Patterns Transfer Object à l'adresse http://java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html.

Utilisation du composant DataSet

En règle générale, vous l'utilisez avec d'autres composants pour manipuler et mettre à jour une source de données : un composant Connector pour la connexion à une source de données externe, des composants d'interface utilisateur pour l'affichage des données issues de la source de données et un composant Resolver pour la conversion des mises à jour du jeu de données au format approprié, pour l'envoi vers la source de données externe. Vous pouvez ensuite utiliser une liaison de données pour relier entre elles les propriétés de ces différents composants.

Le composant DataSet utilise les fonctionnalités des classes de liaison des données. Si vous envisagez d'utiliser le composant DataSet uniquement dans le code ActionScript, sans définir de propriétés via les onglets Liaisons et Schéma de l'inspecteur des composants, importez les classes de liaison des données dans votre fichier FLA et définissez les propriétés nécessaires dans le code. Voir « Disponibilité des classes de liaison des données à l'exécution », à la page 215.

Pour plus d'informations sur la gestion des données de Flash via le composant DataSet, reportez-vous à *Gestion des données* dans *Utilisation de Flash*.

Paramètres du composant DataSet

Vous pouvez définir les paramètres suivants pour le composant DataSet :

itemClassName est une chaîne représentant le nom de la classe d'objet de transfert instanciée à chaque création d'un nouvel élément dans le composant DataSet.

Le composant DataSet utilise des objets de transfert pour représenter les données extraites d'une source de données externe. Si vous laissez ce paramètre vide, le jeu de données crée un objet de transfert anonyme. Si vous lui attribuez une valeur, le jeu de données crée une occurrence de l'objet de transfert à chaque ajout de données.

REMARQUE	Vous devez inclure une référence pleinement qualifiée à cette classe dans votre code pour vous assurer qu'elle est bien compilée dans votre application (par exemple, `private var myItem:my.package.myItem;`).

logChanges est une valeur booléenne ; `true` par défaut. Si ce paramètre est défini sur `true`, le jeu de données garde une trace dans un journal de tous les changements apportés aux données et de tous les appels de méthodes associés aux objets de transfert.

readOnly est une valeur booléenne ; `false` par défaut. Si ce paramètre est défini sur `true`, le jeu de données ne peut pas être modifié.

Vous pouvez écrire du code ActionScript pour utiliser les propriétés, les méthodes et les événements du composant DataSet afin de contrôler ces options et d'autres. Pour plus d'informations, voir « Classe DataSet », à la page 374.

Flux de travaux courant du composant DataSet

Le flux de travaux courant du composant DataSet est décrit ci-dessous.

Pour utiliser un composant DataSet :

1. Assurez-vous que vos paramètres de publication sont bien définis sur ActionScript 2.0.
2. Ajoutez une occurrence du composant DataSet à l'application et nommez-la.
3. Sélectionnez l'onglet Schéma pour le composant DataSet et créez les propriétés de composant représentant les champs persistants du jeu de données.
4. Chargez des données issues d'une source externe dans le composant DataSet. (Pour plus d'informations, consultez la section *Le chargement de données dans le composant DataSet* du guide *Utilisation de Flash*.)
5. Utilisez l'onglet Liaisons de l'inspecteur de composants pour associer les champs de jeu de données aux composants d'interface utilisateur de votre application.

   Lorsque des enregistrements (objets de transfert) sont sélectionnés ou modifiés dans le composant DataSet, les contrôles d'interface utilisateur sont notifiés et mis à jour en conséquence. Les modifications apportées depuis un contrôle d'interface utilisateur sont également notifiées au composant DataSet, puis suivies par le jeu de données et peuvent être extraites par un paquet delta.
6. Pour gérer les données de votre application, appelez les méthodes du composant DataSet.

REMARQUE	En complément de ces étapes, vous pouvez lier le composant DataSet à des composants Connector et Resolver pour disposer d'une solution complète permettant d'accéder aux données d'une source de données externe, de les gérer et de les mettre à jour.

Création d'une application avec le composant DataSet

Généralement, vous utilisez le composant DataSet avec d'autres composants d'interface utilisateur, et souvent avec un composant Connector tel que XMLConnector ou WebServiceConnector. Les éléments du jeu de données sont remplis par le composant Connector ou au moyen de données ActionScript brutes, puis liés aux contrôles d'interface utilisateur (tels que les composants List ou DataGrid).

Le composant DataSet utilise les fonctionnalités des classes de liaison des données. Si vous envisagez d'utiliser le composant DataSet uniquement dans le code ActionScript, sans définir de propriétés via les onglets Liaisons et Schéma de l'inspecteur des composants, importez les classes de liaison des données dans votre fichier FLA et définissez les propriétés nécessaires dans le code. Voir « Disponibilité des classes de liaison des données à l'exécution », à la page 215.

Pour créer une application avec le composant DataSet :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Dans le panneau Composants, faites glisser un composant DataSet jusque sur la scène. Dans l'inspecteur des propriétés, nommez l'occurrence **user_ds**.
3. Faites glisser un composant DataGrid sur la scène et nommez-le **user_dg**.
4. Redimensionnez le composant DataGrid pour qu'il compte environ 300 pixels de large et 100 pixels de haut.
5. Faites glisser un composant Button sur la scène et nommez l'occurrence **next_button**.
6. Dans le scénario, sélectionnez la première image dans le calque 1 et ouvrez le panneau Actions.
7. Ajoutez le code suivant au panneau Actions :

```
var recData_array:Array = [{id:0, firstName:"Mick", lastName:"Jones"},
              {id:1, firstName:"Joe", lastName:"Strummer"},
              {id:2, firstName:"Paul", lastName:"Simonon"}];
user_ds.items = recData_array;
```

   Cette action remplit la propriété `items` de l'objet DataSet avec un tableau d'objets, chacun d'entre eux possédant trois propriétés : `id`, `firstName` et `lastName`.

8. Ajoutez ces trois propriétés et leurs types de données requis au schéma DataSet :
   a. Sélectionnez le composant DataSet sur la scène, ouvrez l'inspecteur des composants et cliquez sur l'onglet Schéma.
   b. Cliquez sur Ajouter une propriété de composant et ajoutez trois nouvelles propriétés, avec les noms de champ `id`, `firstName` et `lastName` et les types de données `Number`, `String` et `String`, respectivement.

   Si vous préférez ajouter les propriétés et leurs types de données requis dans le code, ajoutez la ligne de code suivante dans le panneau Actions au lieu de suivre les étapes a et b ci-dessus :

```
// Ajout des types de schémas requis.
var i:mx.data.types.Str;
var j:mx.data.types.Num;
```

9. Pour lier le contenu du composant DataSet à celui du composant DataGrid, ouvrez l'inspecteur des composants et cliquez sur l'onglet Liaisons.
10. Sélectionnez le composant DataGrid (`user_dg`) sur la scène et cliquez sur le bouton Ajouter une liaison (+) de l'inspecteur des composants.
11. Dans la boîte de dialogue Ajouter une liaison, sélectionnez « dataProvider : Array » et cliquez sur OK.
12. Dans l'inspecteur des composants, double-cliquez sur le champ Lié à.
13. Dans la boîte de dialogue Lié à qui apparaît, sélectionnez « DataSet <user_ds> » dans la colonne Chemin du composant, puis sélectionnez « dataProvider : Array » dans la colonne Emplacement du schéma.
14. Pour lier l'index sélectionné du composant DataSet à celui du composant DataGrid, sélectionnez le composant DataGrid sur la scène et cliquez à nouveau sur le bouton Ajouter une liaison (+) dans l'inspecteur des composants.
15. Dans la boîte de dialogue qui apparaît, sélectionnez « selectedIndex : Number ». Cliquez sur OK.
16. Pour ouvrir la boîte de dialogue Lié à, double-cliquez sur le champ Lié à dans l'inspecteur des composants.
17. Dans le champ Chemin du composant, sélectionnez « DataSet <user_ds> » dans la colonne Chemin du composant et sélectionnez « selectedIndex : Number » dans la colonne Emplacement du schéma.
18. Entrez le code suivant dans le panneau Actions :

```
next_button.addEventListener("click", nextBtnClick);
function nextBtnClick(evt_obj:Object):Void {
  user_ds.next();
}
```

    Ce code utilise la méthode `DataSet.next()` pour naviguer vers l'élément suivant dans la collection d'éléments de l'objet DataSet. Comme vous avez lié la propriété `selectedIndex` de l'objet DataGrid à la même propriété de l'objet DataSet, une modification apportée à l'élément actif dans l'objet DataSet se répercute également sur l'élément actif (sélectionné) dans l'objet DataGrid.
19. Enregistrez le fichier, puis choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

    L'objet DataGrid est rempli avec les éléments spécifiés. Notez que, lorsque vous cliquez sur le bouton, l'élément sélectionné dans l'objet DataGrid change.

Classe DataSet

Héritage MovieClip > DataSet

Nom de classe ActionScript mx.data.components.DataSet

Le composant DataSet vous permet d'utiliser les données en tant que collections d'objets pouvant être indexés, triés, recherchés, filtrés et modifiés.

La fonctionnalité du composant DataSet comprend DataSetIterator, jeu de méthodes pour parcourir et manipuler une collection de données, et DeltaPacket, jeu d'interfaces et de classes pour utiliser des mises à jour d'une collection de données. Dans la plupart des cas, vous n'utilisez pas directement ces classes et interfaces. Vous les utilisez indirectement par l'intermédiaire de méthodes fournies par la classe DataSet.

Méthodes de la classe DataSet

Le tableau suivant présente les méthodes de la classe DataSet.

Méthode	Description
`DataSet.addItem()`	Ajoute l'élément spécifié à la collection.
`DataSet.addItemAt()`	Ajoute un élément au jeu de données à l'emplacement spécifié.
`DataSet.addSort()`	Crée un nouvel affichage trié des éléments de la collection.
`DataSet.applyUpdates()`	Signale que la propriété `deltaPacket` possède une valeur à laquelle vous pouvez accéder par le biais de la liaison de données ou, directement, via ActionScript.
`DataSet.changesPending()`	Indique si la collection présente des modifications en cours qui n'ont pas encore été envoyées dans un paquet delta.
`DataSet.clear()`	Efface tous les éléments de la vue actuelle de la collection.
`DataSet.createItem()`	Renvoie un élément de collection nouvellement initialisé.
`DataSet.disableEvents()`	Arrête l'envoi d'événements DataSet aux écouteurs.
`DataSet.enableEvents()`	Relance l'envoi d'événements DataSet aux écouteurs.
`DataSet.find()`	Localise un élément dans la vue actuelle de la collection.
`DataSet.findFirst()`	Localise la première occurrence d'un élément dans la vue actuelle de la collection.
`DataSet.findLast()`	Localise la dernière occurrence d'un élément dans la vue actuelle de la collection.

Méthode	Description
DataSet.first()	Effectue un déplacement vers le premier élément dans la vue actuelle de la collection.
DataSet.getItemId()	Renvoie l'ID unique de l'élément spécifié.
DataSet.getIterator()	Renvoie un clone de l'itérateur courant.
DataSet.getLength()	Renvoie le nombre d'éléments du jeu de données.
DataSet.hasNext()	Indique si l'itérateur courant est parvenu à la fin de l'affichage de la collection.
DataSet.hasPrevious()	Indique si l'itérateur courant est parvenu au début de l'affichage de la collection.
DataSet.hasSort()	Indique si le tri spécifié existe.
DataSet.isEmpty()	Indique si la collection contient des éléments.
DataSet.last()	Effectue un déplacement jusqu'au dernier élément dans la vue actuelle de la collection.
DataSet.loadFromSharedObj()	Charge toutes les données pertinentes nécessaires à la restauration de la collection DataSet à partir d'un objet partagé.
DataSet.locateById()	Déplace l'itérateur courant vers l'élément possédant l'ID spécifié.
DataSet.next()	Effectue un déplacement jusqu'à l'élément suivant dans la vue actuelle de la collection.
DataSet.previous()	Effectue un déplacement vers l'élément précédent dans la vue actuelle de la collection.
DataSet.removeAll()	Supprime tous les éléments de la collection.
DataSet.removeItem()	Supprime l'élément spécifié de la collection.
DataSet.removeItemAt()	Supprime un élément du jeu de données à l'emplacement spécifié.
DataSet.removeRange()	Supprime les paramètres d'étendue de l'itérateur courant.
DataSet.removeSort()	Supprime le tri spécifié de l'objet DataSet.
DataSet.saveToSharedObj()	Enregistre les données de l'objet DataSet dans un objet partagé.
DataSet.setIterator()	Définit l'itérateur courant pour l'objet DataSet.
DataSet.setRange()	Définit les paramètres d'étendue de l'itérateur courant.

Méthode	Description
DataSet.skip()	Effectue un déplacement vers l'avant ou l'arrière selon un nombre spécifié d'éléments dans la vue actuelle de la collection.
DataSet.useSort()	Le tri spécifié devient actif.

Propriétés de la classe DataSet

Le tableau suivant présente les propriétés de la classe DataSet.

Propriété	Description
DataSet.currentItem	Renvoie l'élément actif dans la collection.
DataSet.dataProvider	Renvoie le fournisseur de données.
DataSet.deltaPacket	Renvoie les modifications apportées à la collection ou affecte des modifications à apporter à la collection.
DataSet.filtered	Indique si des éléments sont filtrés.
DataSet.filterFunc	Fonction définie par l'utilisateur qui permet de filtrer les éléments de la collection.
DataSet.items	Eléments de la collection.
DataSet.itemClassName	Nom de l'objet à créer lors de l'affectation des éléments.
DataSet.length	Spécifie le nombre d'éléments dans la vue actuelle de la collection.
DataSet.logChanges	Indique si les modifications apportées à la collection ou à ses éléments sont enregistrées.
DataSet.properties	Contient les propriétés (champs) de tout objet de transfert de cette collection.
DataSet.readOnly	Indique si la collection peut être modifiée.
DataSet.schema	Spécifie le schéma de la collection au format XML.
DataSet.selectedIndex	Contient l'index de l'élément actif au sein de la collection.

Evénements de la classe DataSet

Le tableau suivant présente les événements de la classe DataSet.

Evénement	Description
`DataSet.addItem`	Diffusé avant l'ajout d'un élément à la collection.
`DataSet.afterLoaded`	Diffusé après l'affectation de la propriété `items`.
`DataSet.calcFields`	Diffusé lorsque les champs calculés doivent être mis à jour.
`DataSet.deltaPacketChanged`	Diffusé lorsque le DeltaPacket de l'objet DataSet a été modifié et qu'il est prêt à être utilisé.
`DataSet.iteratorScrolled`	Diffusé lorsque la position de l'itérateur est modifiée.
`DataSet.modelChanged`	Diffusé lorsque les éléments de la collection ont été modifiés d'une manière quelconque.
`DataSet.newItem`	Diffusé lorsqu'un nouvel objet de transfert est construit par l'objet DataSet, mais avant son ajout à la collection.
`DataSet.removeItem`	Diffusé avant la suppression d'un élément.
`DataSet.resolveDelta`	Diffusé lorsqu'un paquet delta est affecté à l'objet DataSet qui contient des messages.

DataSet.addItem

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.addItem = function (eventObj:Object) {
  // ...
};
dataSetInstance.addEventListener("addItem", listenerObject);
```

Utilisation 2 :

```
on (addItem) {
  // ...
}
```

Description

Evénement généré juste avant l'insertion d'un nouvel enregistrement (objet de transfert) dans cette collection.

Si vous définissez la propriété `result` de l'objet événement sur `false`, l'opération d'ajout est annulée. Si vous la définissez sur `true`, l'opération d'ajout est autorisée.

L'objet événement (*`eventObj`*) possède les propriétés suivantes :

`target` L'objet DataSet qui a généré l'événement.

`type` Chaîne `"addItem"`.

`item` Référence à l'élément de la collection qui doit être ajouté.

`result` Valeur booléenne indiquant si l'élément spécifié doit être ajouté. Par défaut, cette valeur est `true`.

Exemple

Le gestionnaire d'événement `addItem` suivant annule l'ajout du nouvel élément si une fonction définie par l'utilisateur, `userHasAdminPrivs()`, renvoie la valeur `false`. Dans le cas contraire, l'ajout de l'élément est autorisé.

```
function userHasAdminPrivs():Boolean {
  return false; // Modification sur true pour autoriser les insertions.
}

my_ds.addEventListener("addItem", addItemListener);
my_ds.addItem({name:"Bobo", occupation:"clown"});

function addItemListener(evt_obj:Object):Void {
  if (userHasAdminPrivs()) {
    // Autorisation de l'ajout de l'élément.
    evt_obj.result = true;
    trace("Item added");
  } else {
    // Ne pas autoriser l'ajout de l'élément. L'utilisateur ne dispose
    // pas des droits d'administrateur.
    evt_obj.result = false;
    trace("Error, insufficient permissions");
  }
}
```

Voir aussi

`DataSet.removeItem`

DataSet.addItem()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.addItem([*obj*])

Paramètres

obj Objet à ajouter à cette collection. Ce paramètre est facultatif.

Renvoie

Valeur booléenne : `true` si l'élément a été ajouté à la collection, `false` dans le cas contraire.

Description

Méthode qui ajoute l'enregistrement spécifié (objet de transfert) à la collection pour sa gestion. L'élément qui vient d'être ajouté devient l'élément actif du jeu de données. Si aucun paramètre *obj* n'est spécifié, un nouvel objet est créé automatiquement par le biais de `DataSet.createItem()`.

L'emplacement du nouvel élément dans la collection dépend du tri spécifié pour l'itérateur courant. Lorsque aucun tri n'est défini, l'élément est ajouté à la fin de la collection. Si un tri est défini, l'élément est ajouté à la collection en fonction de sa position dans le tri courant.

Pour plus d'informations sur l'initialisation et l'élaboration de l'objet de transfert, reportez-vous à `DataSet.createItem()`.

Exemple

L'exemple suivant utilise `DataSet.addItem()` pour créer un élément et l'ajouter au jeu de données :

```
my_ds.addEventListener("addItem", addItemListener);
my_ds.addItem({name:"Bobo", occupation:"clown"});

function addItemListener(evt_obj:Object):Void {
  trace("adding item");
}
```

L'exemple suivant montre comment vous pouvez accepter ou refuser l'insertion d'un élément dans le composant DataSet en définissant le résultat sur `true` ou `false` dans le gestionnaire pour l'événement `addItem`. Faites glisser un composant DataSet sur la scène et nommez l'occurrence obtenue, `my_ds`. Faites glisser un composant DataGrid sur la scène et nommez l'occurrence, `my_dg`. Faites glisser un composant CheckBox sur la scène et nommez l'occurrence, `my_ch`. Faites glisser un composant Button sur la scène et nommez l'occurrence, `submit_button`. Ajoutez deux propriétés, `name` et `occupation`, au composant DataSet à l'aide de l'onglet Schéma de l'inspecteur de composants. Créez une liaison entre la propriété `my_ds.dataProvider` et la propriété `my_dg.dataProvider` en utilisant le panneau Liaisons de l'inspecteur de composants. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
my_ds.addEventListener("addItem", addItemListener);
submit_button.addEventListener("click", submitListener);
function userHasAdminPrivs():Boolean {
  return my_ch.selected;
}
function addItemListener(evt_obj:Object):Void {
  if (userHasAdminPrivs()) {
    // Autorisation de l'ajout de l'élément.
    evt_obj.result = true;
    trace("Item added");
  } else {
    // Ne pas autoriser l'ajout de l'élément. L'utilisateur ne dispose
    // pas des droits d'administrateur.
    evt_obj.result = false;
    trace("Error, insufficient permissions");
  }
}
function submitListener(evt_obj:Object):Void {
  my_ds.addItem({name:"bobo", occupation:"clown"});
}
```

Voir aussi

`DataSet.createItem()`

DataSet.addItemAt()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.addItemAt(*index*, *item*)

Paramètres

index Nombre supérieur ou égal à 0 qui définit la position à laquelle l'élément doit être inséré. Il s'agit de l'index du nouvel élément.

item Objet contenant les données de l'élément.

Valeur renvoyée

Une valeur booléenne indiquant si l'élément a été ajouté : `true` indique que l'élément a été ajouté et `false` que l'élément existe déjà dans le jeu de données.

Description

Méthode qui ajoute un nouvel élément au jeu de données à l'index spécifié. Les index supérieurs à la longueur du fournisseur de données sont ignorés.

Cette méthode déclenche l'événement `modelChanged` avec le type d'événement `addItem`.

Exemple

L'exemple suivant utilise la méthode `addItemAt()` pour ajouter un élément au composant DataSet à la première position :

```
my_ds.addItem({name:"Milton", years:3});
my_ds.addItem({name:"Mark", years:3});
my_ds.addItem({name:"Sarah", years:1});
my_ds.addItem({name:"Michael", years:2});
my_ds.addItem({name:"Frank", years:2});
my_ds.addItemAt(0, {name:"Bobo", years:1});
```

DataSet.addSort()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.addSort(*name*, *fieldList*, *sortOptions*)

Paramètres

name Chaîne spécifiant le nom du tri.

fieldList Tableau de chaînes spécifiant les noms des champs à trier.

sortOptions Une ou plusieurs des valeurs entières (constantes) suivantes, indiquant les options choisies pour ce tri. Pour séparer plusieurs valeurs, utilisez l'opérateur OR (|) au niveau du bit. Spécifiez une ou plusieurs des valeurs suivantes :

- `DataSetIterator.Ascending` Trie les éléments par ordre croissant. Il s'agit de l'option de tri par défaut lorsque aucune option de tri n'est spécifiée.
- `DataSetIterator.Descending` Trie les éléments par ordre décroissant en fonction des propriétés d'éléments spécifiées.
- `DataSetIterator.Unique` Empêche le tri si des champs ont des valeurs similaires.
- `DataSetIterator.CaseInsensitive` Ignore la casse lors de la comparaison de deux chaînes, au cours du tri. Par défaut, le tri respecte la casse lorsque la propriété triée est une chaîne.

Une exception `DataSetError` est émise lorsque `DataSetIterator.Unique` est spécifié en tant qu'option de tri alors que les données triées ne sont pas uniques, lorsque le nom de tri spécifié a déjà été ajouté ou lorsqu'une propriété spécifiée dans le tableau *fieldList* n'existe pas dans ce jeu de données.

Renvoie

Aucune.

Description

Méthode qui crée un nouveau tri croissant ou décroissant pour l'itérateur actif basé sur les propriétés spécifiées par le paramètre *fieldList*. Flash affecte automatiquement le nouveau tri à l'itérateur courant après sa création, puis le stocke dans la collection de tri pour une utilisation ultérieure.

Exemple

Le code suivant crée un tri, appelé `"nameSort"`, qui réalise un tri par ordre décroissant non sensible à la casse sur le champ `"name"` de l'objet DataSet.

```
import mx.data.components.datasetclasses.DataSetIterator;

my_ds.addItem({name:"Milton", years:3});
my_ds.addItem({name:"mark", years:3});
my_ds.addItem({name:"Sarah", years:1});
my_ds.addItem({name:"michael", years:2});
my_ds.addItem({name:"Frank", years:2});

my_ds.addSort("nameSort", ["name"], DataSetIterator.Descending |
  DataSetIterator.Unique | DataSetIterator.CaseInsensitive);
```

Dans l'exemple suivant, vous pouvez ajouter des données de façon dynamique à un composant DataSet en saisissant les prénoms et les noms dans les occurrences du composant TextInput sur la scène et en appuyant sur le bouton Envoyer. Une fois que vous avez ajouté des éléments au composant DataSet, vous pouvez effacer ce dernier en cliquant sur le bouton Effacer sur la scène. Faites glisser un composant DataGrid sur la scène et nommez l'occurrence, `my_dg`. Faites glisser deux composants Button sur la scène et nommez ces occurrences, `submit_button` et `clear_button`. Faites glisser un composant DataSet sur la scène et nommez l'occurrence, `my_ds`. Faites glisser deux composants TextInput sur la scène et nommez ces occurrences, `firstName_ti` et `lastName_ti`. Faites glisser un composant Alert dans la bibliothèque du document en cours. Ajoutez deux propriétés, `firstName` et `lastName`, au schéma du composant DataSet par le biais de l'onglet Schéma de l'inspecteur de composants. Ensuite, dans l'onglet Liaisons du même inspecteur, ajoutez une liaison de données entre la propriété `dataProvider` du composant DataSet et la propriété `dataProvider` du composant DataGrid. Pour finir, collez le code suivant dans la première image du scénario principal :

```
import mx.controls.Alert;

my_ds.addSort("lastFirst", ["lastName", "firstName"]);

my_dg.enabled = false;
clear_button.enabled = false;
submit_button.label = "Submit";
clear_button.label = "Clear";

my_ds.addEventListener("addItem", addItemListener);
my_ds.addEventListener("modelChanged", modelChangedListener);
submit_button.addEventListener("click", submitListener);
clear_button.addEventListener("click", clearListener);
```

```
function modelChangedListener(evt_obj:Object):Void {
  my_dg.enabled = (evt_obj.target.length > 0);
  clear_button.enabled = my_dg.enabled;
}
function submitListener(evt_obj:Object):Void {
  my_ds.addItem({firstName:firstName_ti.text, lastName:lastName_ti.text});
}
function addItemListener(evt_obj:Object):Void {
  if ((evt_obj.item.firstName.length == 0) || (evt_obj.item.lastName.length
  == 0)) {
    Alert.show("Error, first name or last name cannot be blank.", "Error",
  Alert.OK, _level0);
    evt_obj.result = false;
  } else {
    firstName_ti.text = "";
    lastName_ti.text = "";
  }
}
function clearListener(evt_obj:Object):Void {
  Alert.show("Are you sure you want to clear the data?", "Warning",
  Alert.OK | Alert.CANCEL, _level0, clearConfirmListener);
}
function clearConfirmListener(evt_obj:Object):Void {
  switch (evt_obj.detail) {
  case Alert.OK:
    my_ds.clear();
    break;
  case Alert.CANCEL:
    break;
  }
}
```

Choisissez Contrôle > Tester l'animation pour tester le document dans l'environnement de programmation. Entrez du texte dans les deux occurrences du composant TextInput, puis cliquez sur le bouton Envoyer. Un nouvel élément doit être ajouté à l'occurrence DataGrid. Cliquez sur le bouton Effacer pour afficher une occurrence du composant Alert avec un message de confirmation vous demandant si vous souhaitez effacer le contenu du composant DataGrid. Cliquez sur OK pour effacer la propriété `dataProvider` du composant DataSet (puis la propriété dataProvider du composant DataGrid, en raison de la liaison). Cliquez sur Annuler pour quitter l'occurrence du composant Alert.

Voir aussi

`DataSet.removeSort()`

DataSet.afterLoaded

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.afterLoaded = function (eventObj:Object):Void {
  // ...
};
dataSetInstance.addEventListener("afterLoaded", listenerObject);
```

Utilisation 2 :

```
on (afterLoaded) {
  // ...
}
```

Description

Evénement diffusé immédiatement après l'affectation de la propriété `DataSet.items`.

L'objet événement (*eventObj*) possède les propriétés suivantes :

`target` L'objet DataSet qui a généré l'événement.

`type` Chaîne `"afterLoaded"`.

Exemple

Dans l'exemple suivant, un formulaire appelé `contactForm` (masqué) devient visible dès l'affectation des éléments du jeu de données `contact_ds`.

```
contact_ds.addEventListener("afterLoaded", loadListener);
var loadListener:Object = new Object();
loadListener.afterLoaded = function (evt_obj:Object) {
  if (evt_obj.target == "contact_ds") {
    contactForm.visible = true;
  }
};
```

L'exemple suivant utilise l'événement `afterLoaded` du composant DataSet pour renseigner la propriété `dataProvider` d'un composant List sur la scène. Faites glisser un composant List et un composant DataSet sur la scène et nommez respectivement ces occurrences, `my_list` et `my_ds`. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
my_list.labelField = "name";

var itemsListener:Object = new Object();
itemsListener.afterLoaded = function (evt_obj:Object):Void {
  trace("After loaded");
  my_list.dataProvider = evt_obj.target.items;
}
my_ds.addEventListener("afterLoaded", itemsListener);

var item_array:Array = [{name:"Douglas"}, {name:"Vinnie"},
  {name:"Katherine"}, {name:"David"}];
my_ds.items = item_array;
```

DataSet.applyUpdates()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.applyUpdates()

Renvoie

Aucune.

Description

Méthode ; signale que la propriété `DataSet.deltaPacket` possède une valeur à laquelle vous pouvez accéder par le biais de la liaison de données ou, directement, via ActionScript. Avant l'appel de la méthode, la valeur de la propriété `DataSet.deltaPacket` est `null`. Cette méthode n'a aucun effet si des événements ont été désactivés au moyen de la méthode `DataSet.disableEvents()`.

L'appel de cette méthode crée également un ID de transaction pour la propriété `DataSet.deltaPacket` active et émet un événement `deltaPacketChanged`. Pour plus d'informations, voir `DataSet.deltaPacket`.

Exemple

Le code suivant appelle la méthode `applyUpdates()` sur le DataSet `my_ds`.

```
my_ds.applyUpdates();
```

L'exemple suivant ajoute quatre éléments à l'occurrence du composant `my_ds` DataSet sur la scène et affiche chaque élément dans le niveau supérieur de la propriété `deltaPacket` :

```
my_ds.addItem({name:"Thomas", age:35, gender:"M"});
my_ds.addItem({name:"Orville", age:33, gender:"M"});
my_ds.addItem({name:"Jonathan", age:48, gender:"M"});
my_ds.addItem({name:"Carol", age:31, gender:"F"});

my_ds.applyUpdates();
var i:String;
for (i in my_ds.deltaPacket) {
  trace(i + ":\t" + my_ds.deltaPacket[i]);
}
```

Voir aussi

`DataSet.deltaPacket`

DataSet.calcFields

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.calcFields = function (eventObj:Object):Void {
  // ...
};
dataSetInstance.addEventListener("calcFields", listenerObject);
```

Utilisation 2 :

```
on (calcFields) {
  // ...
}
```

Description

Evénement généré lorsque des valeurs des champs calculés pour l'élément actif dans la collection doivent être déterminées. Un champ est dit calculé lorsque sa propriété Kind est définie sur Calculated dans l'onglet Schéma de l'inspecteur des composants. L'écouteur d'événement `calcFields` que vous créez doit effectuer le calcul requis et définir la valeur du champ calculé.

Cet événement est également appelé lorsque la valeur d'un champ non calculé (champ dont la propriété Kind est définie sur Data dans l'onglet Schéma) est mise à jour.

Pour plus d'informations sur la propriété Kind, reportez-vous à *Types de schéma* dans *Utilisation de Flash*.

ATTENTION	Ne modifiez pas les valeurs des champs non calculés dans cet événement car cela risque de provoquer une « boucle sans fin ». Définissez uniquement les valeurs des champs calculés au sein de l'événement `calcFields`.

DataSet.changesPending()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`dataSetInstance``.changesPending()`

Renvoie

Une valeur booléenne.

Description

Méthode, renvoie `true` si la collection ou l'un de ses éléments a des modifications en attente et que ces dernières n'ont pas encore été envoyées dans un objet DeltaPacket. Dans le cas contraire, elle renvoie `false`.

Exemple

Le code suivant active un bouton Enregistrer les changements (masqué) si la collection DataSet, ou l'un de ses éléments, a fait l'objet de modifications qui n'ont pas encore été soumises dans un objet DeltaPacket.

```
my_ds.addItem({name:"Milton", years:3});
my_ds.addItem({name:"Mark", years:3});
my_ds.addItem({name:"Sarah", years:1});
my_ds.addItem({name:"Michael", years:2});
my_ds.addItem({name:"Frank", years:2});

my_ds.addEventListener("modelChanged", modelChangedListener);
function modelChangedListener(evt_obj:Object):Void {
  if (evt_obj.target.changesPending()) {
    trace("changes pending");
    submitChanges_button.enabled = true;
  }
}
submitChanges_button.enabled = false;
my_ds.addItem({name:"Hal", years:4});
```

DataSet.clear()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.clear()

Renvoie

Aucune.

Description

Méthode qui supprime les éléments de la vue actuelle de la collection. Les éléments qui peuvent être affichés dépendent des paramètres de filtre et d'étendue courants de l'itérateur actuel. Par conséquent, l'appel de cette méthode risque de ne pas supprimer tous les éléments de la collection. Pour supprimer tous les éléments de la collection quel que soit l'affichage de l'itérateur actuel, utilisez `DataSet.removeAll()`.

Si la valeur de `DataSet.logChanges` est définie sur `true` lorsque vous invoquez cette méthode, les entrées « supprimées » sont ajoutées à `DataSet.deltaPacket` pour tous les éléments de la collection.

Exemple

L'exemple suivant supprime tous les éléments de l'affichage en cours de la collection DataSet. La propriété `logChanges` étant définie sur `true`, la suppression de ces éléments est enregistrée.

```
my_ds.addItem({name:"Milton", years:3});
my_ds.addItem({name:"Mark", years:3});
my_ds.addItem({name:"Sarah", years:1});
my_ds.addItem({name:"Michael", years:2});
my_ds.addItem({name:"Frank", years:2});

my_ds.addSort("nameSort", ["name"]);
my_ds.filtered = true;
my_ds.filterFunc = function(item:Object):Boolean {
  return (item.years >= 3);
};
my_ds.logChanges = true;
my_ds.clear(); // Suppression des éléments filtrés du composant DataSet.
my_ds.removeSort("nameSort");
```

Voir aussi

`DataSet.deltaPacket`, `DataSet.logChanges`

DataSet.createItem()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`dataSetInstance``.createItem([`*`itemData`*`])`

Paramètres

`itemData` Données associées à l'élément. Ce paramètre est facultatif.

Renvoie

L'élément qui vient d'être construit.

Description

Méthode : crée un élément qui n'est pas associé à la collection. Vous pouvez spécifier la classe de l'objet créée avec la propriété `DataSet.itemClassName`. Lorsque aucune valeur `DataSet.itemClassName` n'est spécifiée et que le paramètre *`itemData`* est omis, un objet anonyme est construit. Les propriétés de cet objet anonyme sont définies aux valeurs par défaut à partir du schéma actuellement spécifié par `DataSet.schema`.

Lorsque cette méthode est invoquée, tous les écouteurs de l'événement `DataSet.newItem` sont notifiés et en mesure de manipuler l'élément avant qu'il ne soit renvoyé par cette méthode. Les données d'éléments facultatives sont utilisées pour initialiser la classe spécifiée avec la propriété `DataSet.itemClassName` ou exploitées comme élément si `DataSet.itemClassName` est vide.

Une exception `DataSetError` est émise lorsque la classe spécifiée avec la propriété `DataSet.itemClassName` ne peut pas être chargée.

Exemple

```
my_ds.addEventListener("newItem", newItemListener);
function newItemListener(evt_obj:Object):Void {
  trace("new item was added: {name:'" + evt_obj.item.name + "', years:" +
  evt_obj.item.years + "}");
}

my_ds.addItem(my_ds.createItem({name:"Wilson", years:3}));
my_ds.addItem({name:"Tom", years:2});

my_ds.filtered = true;
my_ds.filterFunc = function(item:Object):Boolean {
  return (item.years % 2 == 0);
};
```

Voir aussi

`DataSet.itemClassName`, `DataSet.newItem`, `DataSet.schema`

DataSet.currentItem

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.currentItem

Description

Propriété (lecture seule) ; renvoie l'élément actif dans la collection DataSet ou la valeur `null` si la collection est vide ou si la vue actuelle de la collection de l'itérateur est vide.

Cette propriété offre un accès direct à l'élément au sein de la collection. Les modifications apportées par un accès direct à cet objet ne sont pas suivies (dans la propriété `DataSet.deltaPacket`) et les paramètres de schéma ne sont pas appliqués aux propriétés de cet objet.

Exemple

L'exemple suivant affiche la valeur de la propriété `name` définie dans l'élément actif de l'ensemble de données appelé `customers_ds`.

```
customers_ds.addItem({name:"Milton", years:3});
customers_ds.addItem({name:"Mark", years:3});
customers_ds.addItem({name:"Sarah", years:1});
customers_ds.addItem({name:"Michael", years:2});
customers_ds.addItem({name:"Frank", years:2});

trace(customers_ds.currentItem.name); // Frank
```

DataSet.dataProvider

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.dataProvider

Description

Propriété : fournisseur de données de ce jeu de données. Cette propriété alimente les contrôles d'interface utilisateur, comme les composants List et DataGrid.

Pour plus d'informations sur l'API DataProvider, reportez-vous à « API DataProvider », à la page 333.

Exemple

Le code suivant affecte la propriété `dataProvider` d'un objet DataSet à la propriété correspondante d'un composant DataGrid.

```
my_dg.dataProvider = my_ds.dataProvider;
```

DataSet.deltaPacket

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`dataSetInstance``.deltaPacket`

Description

Propriété qui renvoie un objet DeltaPacket contenant toutes les opérations de modification réalisées sur la collection *`dataSet`* et ses éléments. Cette propriété est `null` tant que `DataSet.applyUpdates()` est appelée sur *`dataSet`*.

Lorsque `DataSet.applyUpdates()` est appelée, un ID de transaction est affecté à l'objet DeltaPacket. Cet ID de transaction permet d'identifier l'objet DeltaPacket sur une boucle de mise à jour à partir du serveur vers le client. Toute affectation ultérieure à la propriété `deltaPacket` via un objet DeltaPacket possédant un ID de transaction correspondant est considérée comme étant la réponse du serveur aux modifications envoyées précédemment. Un objet DeltaPacket possédant un ID correspondant est utilisé pour mettre à jour la collection et rapporter les erreurs spécifiées au sein du paquet.

Les erreurs et les messages du serveur sont rapportés aux écouteurs de l'événement `DataSet.resolveDelta`. Notez que les paramètres `DataSet.logChanges` sont ignorés si un objet DeltaPacket possédant un ID correspondant est affecté à `DataSet.deltaPacket`. Un paquet delta sans ID de transaction correspondant met la collection à jour, comme si les API DataSet étaient utilisées directement. Le cas échéant, des entrées delta supplémentaires peuvent être créées selon ce que le paramètre `DataSet.logChanges` actif de *`dataSet`* et le paquet delta indiquent.

Une exception `DataSetError` est émise lorsqu'un paquet delta est affecté avec un ID de transaction correspondant et que l'un des éléments du paquet delta nouvellement affecté est introuvable dans le paquet d'origine.

Voir aussi

`DataSet.applyUpdates()`, `DataSet.logChanges`, `DataSet.resolveDelta`

DataSet.deltaPacketChanged

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.deltaPacketChanged = function (eventObj:Object):Void {
  // ...
};
dataSetInstance.addEventListener("deltaPacketChanged", listenerObject);
```

Utilisation 2 :

```
on (deltaPacketChanged) {
  // ...
}
```

Description

Evénement ; diffusé lorsque la propriété `deltaPacket` de l'objet DataSet spécifié a été modifiée et qu'elle est prête à être utilisée.

Voir aussi

`DataSet.deltaPacket`

DataSet.disableEvents()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.disableEvents()

Renvoie

Aucune.

Description

Méthode qui désactive des événements pour l'objet DataSet. Lorsque des événements sont désactivés, aucun contrôle d'interface utilisateur (tel qu'un composant DataGrid) n'est mis à jour lorsque des éléments de la collection sont modifiés ou lorsque l'objet DataSet atteint un nouvel élément de la collection.

Pour réactiver des événements, vous devez appeler DataSet.enableEvents(). Pour réactiver la distribution d'événements, la méthode disableEvents() peut être appelée plusieurs fois, mais enableEvents() doit être appelée le même nombre de fois.

Exemple

Dans l'exemple suivant, les événements sont désactivés avant modification des éléments de la collection ; ainsi l'objet DataSet ne peut pas actualiser les contrôles et avoir une incidence sur les performances :

```
my_ds.addEventListener("modelChanged", onModelChanged);
function onModelChanged(evt_obj:Object):Void {
  trace("model changed, DataSet now has " + evt_obj.target.items.length + "
  items");
}
// Désactivation des événements pour le jeu de données.
my_ds.disableEvents();

my_ds.addItem({name:"Apples", price:14});
my_ds.addItem({name:"Bananas", price:8});

trace("Before:");
traceItems();
```

```
my_ds.last();
while(my_ds.hasPrevious()) {
   my_ds.price *= 0.5; // Tout à 50 % !
   my_ds.previous();
}

trace("After:");
traceItems();

// Indication au jeu de données qu'il doit mettre à jour
// les contrôles maintenant.
my_ds.enableEvents();

function traceItems():Void {
  for (var i:Number = 0; i < my_ds.items.length; i++) {
    trace("\t" + my_ds.items[i].name + " - $" + my_ds.items[i].price);
  }
  trace("");
}
```

Voir aussi

`DataSet.enableEvents()`

DataSet.enableEvents()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`dataSetInstance``.enableEvents()`

Renvoie

Aucune.

Description

Méthode ; réactive des événements pour les objets DataSet une fois les événements désactivés par un appel à `DataSet.disableEvents()`. Pour réactiver des événements pour l'objet DataSet, la méthode `enableEvents()` doit être appelée au moins autant de fois que la méthode `disableEvents()`.

Exemple

Dans l'exemple suivant, les événements sont désactivés avant modification des éléments de la collection ; ainsi l'objet DataSet ne peut pas actualiser les contrôles et avoir une incidence sur les performances.

```
my_ds.addEventListener("modelChanged", onModelChanged);
function onModelChanged(evt_obj:Object):Void {
  trace("model changed, DataSet now has " + evt_obj.target.items.length + "
  items");
}
// Désactivation des événements pour le jeu de données.
my_ds.disableEvents();

my_ds.addItem({name:"Apples", price:14});
my_ds.addItem({name:"Bananas", price:8});

trace("Before:");
traceItems();

my_ds.last();
while(my_ds.hasPrevious()) {
   my_ds.price *= 0.5; // Tout à 50 % !
   my_ds.previous();
}

trace("After:");
traceItems();

// Indique au jeu de données qu'il doit mettre à jour
// les contrôles maintenant.
my_ds.enableEvents();

function traceItems():Void {
  for (var i:Number = 0; i < my_ds.items.length; i++) {
    trace("\t" + my_ds.items[i].name + " - $" + my_ds.items[i].price);
  }
  trace("");
}
```

Voir aussi

DataSet.disableEvents()

DataSet.filtered

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.filtered

Description

Propriété : valeur booléenne indiquant si les données de l'itérateur courant sont filtrées. La valeur par défaut est `false`. Lorsque cette propriété est définie sur `true`, la fonction de filtre spécifiée par `DataSet.filterFunc` est appelée pour chaque élément de la collection.

Exemple

Dans l'exemple suivant, le filtre est activé sur l'objet DataSet appelé `employee_ds`. Supposons que chaque enregistrement de la collection possède un champ nommé `empType`. La fonction de filtre suivante renvoie `true` si le champ `empType` dans l'élément actif est défini sur `"management"`. Dans le cas contraire, elle renvoie `false`.

```
employee_ds.filtered = true;
employee_ds.filterFunc = function (item:Object) {
  // Filtrage du personnel d'encadrement.
  return(item.empType != "management");
};
```

L'exemple suivant remplit un composant DataGrid avec un contenu chargé de façon dynamique à l'aide du composant XMLConnector. Lorsqu'un utilisateur clique sur une occurrence du composant CheckBox sur la scène, le contenu du composant DataSet est filtré et mis à jour automatiquement dans le composant DataGrid.

Faites glisser les composants suivants sur la scène et donnez-leur les noms d'occurrence suivants :

- CheckBox (`editorsChoice_ch`)
- DataGrid (`reviews_dg`)
- DataSet (`reviews_ds`)
- XMLConnector (`reviews_xmlconn`)

Téléchargez une copie du document XML suivant et enregistrez-la sur votre disque dur local :`http://www.helpexamples.com/flash/xml/reviews.xml`. Ce document XML sera chargé de façon dynamique à l'aide du composant XMLConnector, mais vous utiliserez la copie locale pour importer le schéma XML dans le composant DataSet. Sélectionnez l'occurrence XMLConnector sur la scène et cliquez sur l'onglet Schéma dans l'inspecteur de composants. Sélectionnez la propriété `results` et cliquez sur le bouton « Import a schema from a sample XML file (Importer un schéma à partir d'un fichier d'exemple XML) ». Sélectionnez le document XML que vous avez téléchargé sur votre disque dur local, puis cliquez sur Ouvrir. Le schéma du document XML doit être importé dans le composant DataSet. L'occurrence du composant XMLConnector `reviews_xmlconn` étant encore sélectionnée sur la scène, ajoutez une liaison entre le tableau `reviews_xmlconn.results.reviews.review` et la propriété `dataProvider` de l'occurrence DataSet `reviews_ds`. Définissez la direction de la liaison des données sur « out » dans l'onglet Liaisons. Sélectionnez l'occurrence DataSet `reviews_ds` sur la scène et ajoutez une autre liaison pour la propriété `dataProvider`. L'occurrence `reviews_ds` étant sélectionnée, cliquez sur l'onglet Liaisons dans l'inspecteur de composants. Cliquez sur le bouton Ajouter une liaison, puis ajoutez une liaison entre la propriété `dataProvider` du composant DataSet et la propriété `dataProvider` de l'occurrence DataGrid `reviews_dg`.

Ajoutez le code suivant à l'image 1 du scénario principal :

```
editorsChoice_ch.label = "Editor's Choice";

reviews_xmlconn.direction = "receive";
reviews_xmlconn.multipleSimultaneousAllowed = false;
reviews_xmlconn.URL = "http://www.helpexamples.com/flash/xml/reviews.xml";
reviews_xmlconn.trigger();

reviews_dg.setSize(320, 240);
reviews_dg.addColumn("name");
reviews_dg.addColumn("rating");
reviews_dg.addColumn("reviewDate");
reviews_dg.getColumnAt(0).width = 100;
reviews_dg.getColumnAt(1).width = 100;
reviews_dg.getColumnAt(2).width = 100;
reviews_dg.setStyle("alternatingRowColors", [0xFFFFFF, 0xF6F6F6]);

editorsChoice_ch.addEventListener("click", editorsChoiceListener);
function editorsChoiceListener(evt_obj:Object):Void {
  reviews_ds.filtered = evt_obj.target.selected;
  reviews_ds.filterFunc = function(item:Object):Boolean {
    return (item.editorsChoice == 1);
  };
}
```

Sélectionnez Contrôle > Tester l'animation. Par défaut, le composant DataGrid affiche toutes les révisions du document XML externe. Le fait de cliquer sur le composant Editor's Choice CheckBox oblige le composant DataSet à filtrer et à afficher uniquement les révisions marquées comme choix de l'éditeur.

Voir aussi

`DataSet.filterFunc`

DataSet.filterFunc

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
dataSetInstance.filterFunc = function (item:Object):Boolean {
  // Renvoi de true|false
};
```

Description

Propriété qui désigne la fonction qui détermine les éléments inclus dans la vue actuelle de la collection. Lorsque `DataSet.filtered` est définie sur `true`, la fonction affectée à cette propriété est appelée pour chaque enregistrement (objet de transfert) de la collection. Pour chaque élément transmis à la fonction, elle doit renvoyer `true` si l'élément est à inclure dans la vue actuelle ou `false` dans le cas contraire.

Lors d'un changement de la fonction de filtre sur le jeu de données, vous devez définir la propriété `filtered` sur `false`, puis de nouveau sur `true` pour que l'événement `modelChanged` approprié soit généré. La modification de la propriété `filterFunc` ne génère pas l'événement.

De même, si un filtre est déjà en place lors du chargement des données dans (`modelChanged` ou `updateAll`), il n'est pas appliqué avant que la valeur de `filtered` ne soit définie sur `false`, puis de nouveau sur `true`.

Exemple

Dans l'exemple suivant, le filtre est activé sur l'objet DataSet appelé `employee_ds`. La fonction de filtre spécifiée renvoie `true` si le champ `empType` de chaque élément est défini sur `"management"`. Dans le cas contraire, elle renvoie `false`.

```
employee_ds.filtered = true;
employee_ds.filterFunc = function (item:Object):Boolean {
  // Filtrage du personnel d'encadrement.
  return(item.empType != "management");
};
```

Dans l'exemple suivant, vous filtrez le contenu d'un composant DataSet en fonction de l'élément sélectionné dans un composant ComboBox. Le composant ComboBox vous permet d'effectuer une sélection parmi tout le monde, les hommes uniquement ou les femmes uniquement.

Faites glisser un composant DataSet, DataGrid et ComboBox sur la scène et nommez respectivement ces occurrences, `my_ds`, `data_dg` et `filter_cb`. Sélectionnez l'occurrence du DataSet `my_ds` sur la scène et ajoutez les deux nouvelles propriétés : `name` et `gender`. Dans le panneau Liaisons de l'inspecteur de composants, créez une liaison de données entre la propriété du composant DataSet `dataProvider` et la propriété `dataProvider` du composant DataGrid. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
my_ds.dataProvider = new Array({name:"Charles", gender:"M"}, {name:"Buddy",
  gender:"M"}, {name:"Walter", gender:"M"}, {name:"Ellen", gender:"F"},
  {name:"Jamie", gender:"F"}, {name:"Sarah", gender:"F"}, {name:"Adam",
  gender:"M"});
my_ds.addSort("nameSort", ["name"]);
my_ds.addSort("genderSort", ["gender", "name"]);
my_ds.useSort("genderSort");

filter_cb.dataProvider = [{label:"All", value:""} , {label:"Male only",
  value:"M"}, {label:"Female only", value:"F"}];
filter_cb.addEventListener("change", filterListener);
function filterListener(evt_obj:Object):Void {
  var selItem:Object = evt_obj.target.selectedItem;
  if (selItem.value.length == 0) {
    my_ds.filtered = false;
  } else {
    my_ds.filtered = true;
    my_ds.filterFunc = function(item:Object):Boolean {
      return (item.gender == selItem.value);
    }
  }
}
```

L'exemple suivant remplit un composant DataGrid avec un contenu chargé de façon dynamique à l'aide du composant XMLConnector. Lorsqu'un utilisateur clique sur une occurrence du composant CheckBox sur la scène, le contenu du composant DataSet est filtré et mis à jour automatiquement dans le composant DataGrid.

Faites glisser les composants suivants sur la scène et donnez-leur les noms d'occurrence suivants :

- CheckBox (`editorsChoice_ch`)
- DataGrid (`reviews_dg`)
- DataSet (`reviews_ds`)
- XMLConnector (`reviews_xmlconn`)

Téléchargez une copie du document XML suivant, puis enregistrez-la sur votre disque dur local : `www.helpexamples.com/flash/xml/reviews.xml`. Ce document XML est chargé de façon dynamique à l'aide du composant XMLConnector. Utilisez ensuite cette copie locale pour importer le schéma XML dans le composant DataSet. Sélectionnez l'occurrence XMLConnector sur la scène, puis cliquez sur l'onglet Schéma dans l'inspecteur de composants. Sélectionnez la propriété `results`, puis cliquez sur Importer un schéma d'un exemple de fichier XML. Sélectionnez le document XML que vous avez téléchargé sur votre disque dur local, puis cliquez sur Ouvrir. Le schéma du document XML doit être importé dans le composant DataSet. L'occurrence du composant XMLConnector `reviews_xmlconn` étant encore sélectionnée sur la scène, ajoutez une liaison entre le tableau `reviews_xmlconn.results.reviews.review` et la propriété `dataProvider` de l'occurrence DataSet `reviews_ds`. Définissez la direction de la liaison des données sur « out » dans l'onglet Liaisons. Sélectionnez l'occurrence DataSet `reviews_ds` sur la scène, puis ajoutez une autre liaison pour la propriété `dataProvider`. L'occurrence `reviews_ds` étant sélectionnée, cliquez sur l'onglet Liaisons dans l'inspecteur de composants. Cliquez sur le bouton Ajouter une liaison, puis ajoutez une liaison entre la propriété `dataProvider` du composant DataSet et la propriété `dataProvider` de l'occurrence DataGrid `reviews_dg`.

Ajoutez le code suivant à l'image 1 du scénario principal :

```
editorsChoice_ch.label = "Editor's Choice";

reviews_xmlconn.direction = "receive";
reviews_xmlconn.multipleSimultaneousAllowed = false;
reviews_xmlconn.URL = "http://www.helpexamples.com/flash/xml/reviews.xml";
reviews_xmlconn.trigger();

reviews_dg.setSize(320, 240);
reviews_dg.addColumn("name");
reviews_dg.addColumn("rating");
reviews_dg.addColumn("reviewDate");
reviews_dg.getColumnAt(0).width = 100;
```

```
reviews_dg.getColumnAt(1).width = 100;
reviews_dg.getColumnAt(2).width = 100;
reviews_dg.setStyle("alternatingRowColors", [0xFFFFFF, 0xF6F6F6]);

editorsChoice_ch.addEventListener("click", editorsChoiceListener);
function editorsChoiceListener(evt_obj:Object):Void {
  reviews_ds.filtered = evt_obj.target.selected;
  reviews_ds.filterFunc = function(item:Object):Boolean {
    return (item.editorsChoice == 1);
  };
}
```

Sélectionnez Contrôle > Tester l'animation. Par défaut, le composant DataGrid affiche toutes les révisions du document XML externe. Le fait de cliquer sur le composant Editor's Choice CheckBox oblige le composant DataSet à filtrer et à afficher uniquement les révisions marquées comme choix de l'éditeur.

Voir aussi

`DataSet.filtered`

DataSet.find()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`dataSetInstance``.find(`*`searchValues`*`)`

Paramètres

`searchValues` Tableau contenant une ou plusieurs valeurs de champ pouvant être localisées dans le tri en cours.

Renvoie

`true` si les valeurs sont trouvées, `false` dans le cas contraire.

Description

Méthode qui recherche dans la vue actuelle de la collection un élément avec des valeurs de champ spécifiées par *`searchValues`*. Les éléments qui se trouvent dans la vue actuelle dépendent des paramètres de filtre et d'étendue courants. Lorsqu'un élément est trouvé, il devient l'élément actif dans l'objet DataSet.

Les valeurs spécifiées par *searchValues* doivent être dans le même ordre que la liste de champs spécifiée par le tri courant (voir l'exemple ci-dessous).

Si le tri courant n'est pas unique, l'objet de transfert trouvé est non déterminant. Si vous souhaitez trouver la première ou la dernière occurrence d'un objet de transfert dans un tri qui n'est pas unique, utilisez `DataSet.findFirst()` ou `DataSet.findLast()`.

La conversion spécifiée pour les données repose sur le type de champ sous jacent. Par exemple, si vous spécifiez `["05-02-02"]` comme valeur de recherche, le champ de date sous-jacent est utilisé pour convertir la valeur en utilisant la méthode `DataType.setAsString()` de la date. Si vous spécifiez `[new Date().getTime()]`, la méthode `DataType.setAsNumber()` de la date est utilisée.

Exemple

L'exemple suivant recherche un élément dans la collection actuelle dont les champs `name` et `id` contiennent respectivement les valeurs `"Bobby"` et 105. Si l'élément est trouvé, `DataSet.getItemId()` est utilisée pour obtenir son identificateur unique dans la collection, et `DataSet.locateById()` permet de positionner l'itérateur actuel sur cet élément.

```
var studentID:String = null;
student_ds.addSort("id", ["name","id"]);
// Localisation de l'objet de transfert identifié par « Bobby » et 105.
// Notez que l'ordre des champs de recherche correspond aux champs
// spécifiés dans la méthode addSort().
if (student_ds.find(["Bobby", 105])) {
  studentID = student_ds.getItemId();
}
// A présent, utilisez la méthode locateByID() pour positionner l'itérateur
// sur l'élément de la collection dont l'ID correspond à studentID.
if (studentID != null) {
  student_ds.locateById(studentID);
}
```

Voir aussi

`DataSet.applyUpdates()`, `DataSet.getItemId()`, `DataSet.locateById()`

DataSet.findFirst()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.findFirst(*searchValues*)

Paramètres

searchValues Tableau contenant une ou plusieurs valeurs de champ pouvant être localisées dans le tri en cours.

Renvoie

`true` si les éléments sont trouvés, `false` dans le cas contraire.

Description

Méthode qui recherche dans la vue actuelle de la collection le premier élément avec les valeurs de champ spécifiées par *searchValues*. Les éléments qui se trouvent dans l'affichage en cours dépendent des paramètres de filtre et d'étendue courants.

Les valeurs spécifiées par *searchValues* doivent être dans le même ordre que la liste de champs spécifiée par le tri courant (voir l'exemple ci-dessous).

La conversion spécifiée pour les données repose sur le type de champ sous jacent. Par exemple, si la valeur de recherche spécifiée est `["05-02-02"]`, le champ de date sous-jacent est utilisé pour convertir la valeur à l'aide de la méthode `setAsString()` de la date. Si la valeur spécifiée est `[new Date().getTime()]`, la méthode `setAsNumber()` de la date est utilisée.

Exemple

L'exemple suivant utilise `DataSet.find()` pour rechercher un élément dans la collection actuelle dont les champs `name` et `id` contiennent respectivement les valeurs `"Bobby"` et 105. Si l'élément est trouvé, `DataSet.getItemId()` sert à obtenir son identificateur unique dans la collection et `DataSet.locateById()`, à positionner l'itérateur courant sur cet élément.

Pour tester cet exemple, faites glisser un composant DataSet sur la scène et nommez l'occurrence, `student_ds`. Ajoutez deux propriétés, `name` (type de données : String) et `id` (type de données : Number) au composant DataSet à l'aide de l'onglet Schéma de l'inspecteur de composants. Si vous ne possédez pas encore de copie du clip compilé `DataBindingClasses` dans votre bibliothèque, faites glisser une copie du clip compilé de la bibliothèque Classes (Fenêtre > Bibliothèques communes > Classes). Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
student_ds.addItem({name:"Barry", id:103});
student_ds.addItem({name:"Bobby", id:105});
student_ds.addItem({name:"Billy", id:107});

trace("Before find() > " + student_ds.currentItem.name); // Billy

var studentID:String;
student_ds.addSort("id", ["name","id"]);
if (student_ds.find(["Bobby", 105])) {
  studentID = student_ds.getItemId();
  student_ds.locateById(studentID);
  trace("After find() > " + student_ds.currentItem.name); // Bobby
} else {
  trace("We lost Billy!");
}
```

Voir aussi

`DataSet.applyUpdates()`, `DataSet.getItemId()`, `DataSet.locateById()`

DataSet.findLast()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`dataSetInstance``.findLast(`*`searchValues`*`)`

Paramètres

`searchValues` Tableau contenant une ou plusieurs valeurs de champ pouvant être localisées dans le tri en cours.

Renvoie

`true` si les éléments sont trouvés, `false` dans le cas contraire.

Description

Méthode qui recherche dans la vue actuelle de la collection le dernier élément ayant les valeurs de champ spécifiées par *`searchValues`*. Les éléments qui se trouvent dans l'affichage en cours dépendent des paramètres de filtre et d'étendue courants.

Les valeurs spécifiées par *`searchValues`* doivent être dans le même ordre que la liste de champs spécifiée par le tri courant (voir l'exemple ci-dessous).

La conversion spécifiée pour les données repose sur le type de champ sous jacent. Par exemple, si la valeur de recherche spécifiée est `["05-02-02"]`, le champ de date sous-jacent est utilisé pour convertir la valeur à l'aide de la méthode `setAsString()` de la date. Si la valeur spécifiée est `[new Date().getTime()]`, la méthode `setAsNumber()` de la date est utilisée.

Exemple

L'exemple suivant recherche le dernier élément de la collection actuelle dont les champs `name` et `age` contiennent `"Bobby"` et `"13"`. Si l'élément est trouvé, `DataSet.getItemId()` sert à obtenir son identificateur unique dans la collection, et `DataSet.locateById()` à positionner l'itérateur actuel sur cet élément.

```
var studentID:String = null;
student_ds.addSort("nameAndAge", ["name", "age"]);
// Localisation du dernier objet de transfert ayant les valeurs spécifiées.
// Notez que l'ordre des champs de recherche correspond aux champs
// spécifiés dans la méthode addSort().
if (student_ds.findLast(["Bobby", "13"])) {
  studentID = student_ds.getItemId();
}

// A présent, utilisez la méthode locateByID() pour positionner l'itérateur
// sur l'élément de la collection dont l'ID correspond à studentID.
if (studentID != null) {
  student_ds.locateById(studentID);
}
```

Voir aussi

`DataSet.applyUpdates()`, `DataSet.getItemId()`, `DataSet.locateById()`

DataSet.first()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.first()

Renvoie

Aucune.

Description

Méthode qui transforme le premier élément de la vue actuelle de la collection en élément actif. Les éléments qui se trouvent dans la vue actuelle dépendent des paramètres de filtre et d'étendue courants.

Exemple

Le code suivant positionne le jeu de données `inventory_ds` au niveau du premier élément dans sa collection, puis affiche la valeur de la propriété `price` que contient cet élément en utilisant la propriété `DataSet.currentItem`.

```
inventory_ds.first();
trace("The price of the first item is:" + inventory_ds.currentItem.price);
```

L'exemple suivant est répété sur tous les éléments de l'affichage actuel de la collection (en commençant par le début), puis calcule la propriété `price` de chaque élément.

```
my_ds.addItem({name:"item a", price:16});
my_ds.addItem({name:"item b", price:9});

my_ds.first();
while (my_ds.hasNext()) {
  my_ds.currentItem.price *= 0.5; // Tout à 50 % !
  my_ds.next();
}

for (var i in my_ds.items) {
  trace(my_ds.items[i].name + ": " + my_ds.items[i].price);
}
```

Voir aussi

`DataSet.last()`

DataSet.getItemId()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.getItemId([*index*])

Paramètres

index Nombre spécifiant l'élément de la vue actuelle dont l'ID est recherché. Ce paramètre est facultatif.

Renvoie

Une chaîne.

Description

Méthode qui renvoie l'identifiant de l'élément actif de la collection ou celui de l'élément spécifié par l'*index*. Cet identifiant n'est unique que dans cette collection et il est affecté automatiquement par `DataSet.addItem()`.

Exemple

Le code suivant obtient l'ID unique de l'élément actuel dans la collection et l'affiche dans le panneau Sortie.

```
var itemNo:String = my_ds.getItemId();
trace("Employee id(" + itemNo + ")");
```

L'exemple suivant utilise `DataSet.find()` pour rechercher un élément dans la collection actuelle dont les champs `name` et `id` contiennent respectivement les valeurs `"Bobby"` et 105. Si l'élément est trouvé, `DataSet.getItemId()` sert à obtenir son identificateur unique dans la collection, et `DataSet.locateById()` à positionner l'itérateur courant sur cet élément.

Pour tester cet exemple, faites glisser un composant DataSet sur la scène et nommez l'occurrence, `student_ds`. Ajoutez deux propriétés, `name` (type de données : String) et `id` (type de données : Number), au composant DataSet à l'aide de l'onglet Schéma de l'inspecteur de composants. Si vous ne possédez pas encore de copie du clip compilé `DataBindingClasses` dans votre bibliothèque, faites glisser une copie du clip compilé de la bibliothèque Classes (Fenêtre > Bibliothèques communes > Classes). Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
student_ds.addItem({name:"Barry", id:103});
student_ds.addItem({name:"Bobby", id:105});
student_ds.addItem({name:"Billy", id:107});

trace("Before find() > " + student_ds.currentItem.name); // Billy

var studentID:String;
student_ds.addSort("id", ["name","id"]);
if (student_ds.find(["Bobby", 105])) {
  studentID = student_ds.getItemId();
  student_ds.locateById(studentID);
  trace("After find() > " + student_ds.currentItem.name); // Bobby
} else {
  trace("We lost Billy!");
}
```

Voir aussi

DataSet.addItem()

DataSet.getIterator()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.getIterator()

Renvoie

Objet ValueListIterator.

Description

Méthode qui renvoie un nouvel itérateur pour cette collection. Cet itérateur est le clone de l'itérateur courant, notamment de sa position courante dans la collection. Cette méthode s'adresse avant tout aux utilisateurs expérimentés qui souhaitent avoir accès à plusieurs affichages simultanés de la même collection.

Exemple

L'exemple suivant utilise `DataSet.find()` pour rechercher un élément dans la collection actuelle dont le champ `name` contient la valeur `"Bobby"`. Même si l'itérateur `myIterator` pointe vers l'enregistrement de Bobby, l'itérateur principal de `student_ds` pointe toujours vers le dernier enregistrement, Billy.

Pour tester cet exemple, faites glisser un composant DataSet sur la scène et nommez l'occurrence, `student_ds`. Ajoutez deux propriétés, `name` (type de données : String) et `id` (type de données : Number), au composant DataSet à l'aide de l'onglet Schéma de l'inspecteur de composants. Si vous ne possédez pas encore de copie du clip compilé `DataBindingClasses` dans votre bibliothèque, faites glisser une copie du clip compilé de la bibliothèque Classes (Fenêtre > Bibliothèques communes > Classes). Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
import mx.data.to.ValueListIterator;

student_ds.addItem({name:"Barry", id:103});
student_ds.addItem({name:"Bobby", id:105});
student_ds.addItem({name:"Billy", id:107});

var myIterator:ValueListIterator = student_ds.getIterator();
myIterator.sortOn(["name"]);
myIterator.find({name:"Bobby"}).id = "999";

trace(student_ds.currentItem.name + " [" + student_ds.currentItem.id +
  "]");
  // Billy [107]

student_ds.addSort("id", ["name", "id"]);
if (student_ds.find({name:"Bobby", id:999})) {
  student_ds.locateById(student_ds.getItemId());
  trace(student_ds.currentItem.name + " [" + student_ds.currentItem.id +
  "]");
    // Bobby [999]
} else {
  trace("We lost Billy!");
}
```

DataSet.getLength()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.getLength()

Renvoie

Nombre d'éléments du jeu de données.

Description

Méthode qui renvoie le nombre d'éléments du jeu de données.

Exemple

L'exemple suivant appelle `getLength()` :

```
//...
var my_ds:mx.data.components.DataSet;
my_ds = _parent.thisShelf.compactDiscs_ds;
trace ("Data set size is: " + my_ds.getLength());
//...
```

DataSet.hasNext()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.hasNext()

Renvoie

Valeur booléenne.

Description

Méthode qui renvoie `false` si l'itérateur courant est à la fin de l'affichage de la collection, ou `true` dans le cas contraire.

Exemple

L'exemple suivant se répète sur tous les éléments de l'affichage actuel de la collection (en commençant par le début), puis calcule la propriété `price` de chaque élément.

```
my_ds.addItem({name:"item a", price:16});
my_ds.addItem({name:"item b", price:9});

my_ds.first();
while (my_ds.hasNext()) {
  my_ds.currentItem.price *= 0.5; // Tout à 50 % !
  my_ds.next();
}

for (var i in my_ds.items) {
  trace(my_ds.items[i].name + ": " + my_ds.items[i].price);
}
```

Voir aussi

`DataSet.currentItem`, `DataSet.first()`, `DataSet.next()`

DataSet.hasPrevious()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`dataSetInstance``.hasPrevious()`

Renvoie

Une valeur booléenne.

Description

Méthode qui renvoie `false` si l'itérateur courant est au début de l'affichage de la collection, ou `true` dans le cas contraire.

Exemple

L'exemple suivant se répète sur tous les éléments de la vue actuelle de la collection (en commençant par le dernier élément), puis calcule la propriété `price` de chaque élément :

```
my_ds.addItem({name:"item a", price:16});
my_ds.addItem({name:"item b", price:9});

my_ds.last();
while (my_ds.hasPrevious()) {
  my_ds.currentItem.price *= 0.5; // Tout à 50 % !
  my_ds.previous();
}

for (var i in my_ds.items) {
  trace(my_ds.items[i].name + ": " + my_ds.items[i].price);
}
```

Voir aussi

`DataSet.currentItem`, `DataSet.skip()`, `DataSet.previous()`

DataSet.hasSort()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`dataSetInstance``.hasSort(`*`sortName`*`)`

Paramètres

`sortName` Chaîne qui contient le nom d'un tri créé avec `DataSet.addSort()`.

Renvoie

Une valeur booléenne.

Description

Méthode qui renvoie `true` si le tri spécifié par *`sortName`* existe, ou `false` dans le cas contraire.

Exemple

Le code suivant vérifie qu'un tri nommé « nameSort » existe. Si le tri existe déjà, il devient le tri actuel via la méthode `DataSet.useSort()`. Lorsqu'il n'existe pas encore de tri par ce nom, il est créé au moyen de la méthode `DataSet.addSort()`. Pour tester cet exemple, faites glisser un composant DataSet et un composant List sur la scène et nommez respectivement les occurrences `my_ds` et `my_list`. Dans l'onglet Liaisons de l'inspecteur de composants, ajoutez une liaison entre la propriété du composant DataSet `dataProvider` et la propriété `dataProvider` du composant List. Créez deux propriétés dans le schéma `my_ds` du composant DataSet à l'aide de l'onglet Schéma de l'inspecteur de composants : `name` (type de données : String) et `years` (type de données : Number). Ajoutez le code suivant sur l'image 1 du scénario principal :

```
import mx.data.components.datasetclasses.DataSetIterator;
my_list.labelField = "name";

my_ds.addItem({name:"Milton", years:3});
my_ds.addItem({name:"Mark", years:3});
my_ds.addItem({name:"Sarah", years:1});
my_ds.addItem({name:"Michael", years:2});
my_ds.addItem({name:"Frank", years:2});

if (my_ds.hasSort("nameSort")) {
  my_ds.useSort("nameSort");
} else {
  my_ds.addSort("nameSort", ["name"], DataSetIterator.Descending);
}
```

Voir aussi

`DataSet.addSort()`, `DataSet.applyUpdates()`, `DataSet.useSort()`

DataSet.isEmpty()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`dataSetInstance``.isEmpty()`

Renvoie

Valeur booléenne.

Description

Méthode ; renvoie `true` si l'objet DataSet spécifié ne contient aucun objet (si *`dataSet`*`.length == 0`).

Exemple

Le code suivant désactive le bouton Supprimer l'enregistrement (masqué) lorsque l'objet DataSet auquel il s'applique est vide :

```
if (my_ds.isEmpty()) {
  delete_button.enabled = false;
}
```

Voir aussi

`DataSet.length`

DataSet.items

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`dataSetInstance``.items`

Description

Propriété ; tableau d'éléments gérés par *`my_ds`*.

Exemple

L'exemple suivant affecte un tableau d'objets à la propriété `items` d'un objet DataSet :

```
var recData:Array = [{id:0, firstName:"Mick", lastName:"Jones"},
                     {id:1, firstName:"Joe", lastName:"Strummer"},
                     {id:2, firstName:"Paul", lastName:"Simonon"}];
my_ds.items = recData;
```

DataSet.itemClassName

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
dataSetInstance.itemClassName
```

Description

Propriété : chaîne indiquant le nom de la classe qui doit être créée lorsque des éléments sont ajoutés à la collection. La classe que vous spécifiez doit implémenter l'interface TransferObject, comme indiqué ci-dessous.

```
interface mx.data.to.TransferObject {
  function clone():Object;
  function getPropertyData():Object;
  function setPropertyData(propData:Object):Void;
}
```

Vous pouvez également définir cette propriété dans l'inspecteur Propriétés.

Pour que cette classe soit disponible à l'exécution, vous devez également inclure une référence pleinement qualifiée à cette classe au sein du code de votre fichier SWF, comme dans le code suivant :

```
var myItem:my.package.myItem;
```

Une exception `DataSetError` est émise si vous essayez de modifier la valeur de cette propriété après le chargement du tableau `DataSet.items`.

Pour plus d'informations, voir « Interface TransferObject », à la page 1281.

DataSet.iteratorScrolled

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.iteratorScrolled = function (eventObj:Object) {
  // ...
};
dataSetInstance.addEventListener("iteratorScrolled", listenerObject);
```

Utilisation 2 :

```
on (iteratorScrolled) {
  // ...
}
```

Description

Evénement généré dès que l'itérateur courant atteint un nouvel élément de la collection.

L'objet événement (*eventObj*) possède les propriétés suivantes :

`target` L'objet DataSet qui a généré l'événement.

`type` Chaîne `"iteratorScrolled"`.

`scrolled` Nombre qui spécifie le nombre d'éléments que l'itérateur a parcouru. Les valeurs positives indiquent que l'itérateur a avancé dans la collection, et les valeurs négatives qu'il a reculé.

Exemple

Dans l'exemple suivant, la barre d'état d'une application (masquée) est mise à jour lorsque la position de l'itérateur actuel change :

```
my_ds.addItem({name:"Milton", years:3});
my_ds.addItem({name:"Mark", years:3});
my_ds.addItem({name:"Sarah", years:1});
my_ds.addItem({name:"Michael", years:2});
my_ds.addItem({name:"Frank", years:2});

my_ds.addEventListener("iteratorScrolled", iteratorScrolledListener);

my_ds.first(); // Déclenchement de l'événement iteratorScrolled.

function iteratorScrolledListener(evt_obj:Object):Void {
  trace("Défilement de l'itérateur.");
}
```

DataSet.last()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.last()

Renvoie

Aucune.

Description

Méthode qui transforme le dernier élément de la vue actuelle de la collection en élément actif.

Exemple

Le code suivant, associé à un composant Button, permet d'atteindre le dernier élément de la collection DataSet :

```
function goLast(evt_obj:obj):Void {
  inventory_ds.last();
}
goLast_button.addEventListener("click", goLast);
```

L'exemple suivant est répété sur tous les éléments de l'affichage actuel de la collection (en commençant par le dernier élément), puis calcule la propriété `price` de chaque élément :

```
my_ds.addItem({name:"item a", price:16});
my_ds.addItem({name:"item b", price:9});

my_ds.last();
while (my_ds.hasPrevious()) {
  my_ds.currentItem.price *= 0.5; // Tout à 50 % !
  my_ds.previous();
}

for (var i in my_ds.items) {
  trace(my_ds.items[i].name + ": " + my_ds.items[i].price);
}
```

Voir aussi

DataSet.first()

DataSet.length

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.length

Description

Propriété (lecture seule) qui spécifie le nombre d'éléments présents dans la vue actuelle de la collection. Le nombre d'éléments pouvant être affichés dépend des paramètres de filtre et d'étendue courants.

Exemple

Dans l'exemple suivant, les événements sont désactivés avant modification des éléments de la collection, ainsi l'objet DataSet n'aura aucune incidence sur les performances en essayant d'actualiser les contrôles :

```
my_ds.addEventListener("modelChanged", onModelChanged);
function onModelChanged(evt_obj:Object):Void {
  trace("model changed, DataSet now has " + evt_obj.target.length + "
  items");
}
// Désactivation des événements pour le jeu de données.
my_ds.disableEvents();

my_ds.addItem({name:"Apples", price:14});
my_ds.addItem({name:"Bananas", price:8});

trace("Before:");
traceItems();

my_ds.last();
while(my_ds.hasPrevious()) {
   my_ds.price *= 0.5; // Tout à 50 % !
   my_ds.previous();
}

trace("After:");
traceItems();
```

```
// Indication au jeu de données qu'il doit mettre à jour
// les contrôles maintenant.
my_ds.enableEvents();

function traceItems(label:String):Void {
  for (var i:Number = 0; i < my_ds.length; i++) {
    trace("\t" + my_ds.items[i].name + " - $" + my_ds.items[i].price);
  }
  trace("");
}
```

DataSet.loadFromSharedObj()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.loadFromSharedObj(*objName*, [*localPath*])

Paramètres

objName Chaîne spécifiant le nom de l'objet partagé à récupérer. Le nom peut inclure des barres obliques (par exemple, « travail/adresses »). Les espaces et les caractères suivants ne sont pas autorisés dans le nom spécifié :

~ % & \ ; : " ' , < > ? #

localPath Paramètre de chaîne facultatif qui précise le chemin complet ou partiel du fichier SWF qui a créé l'objet partagé. La chaîne est utilisée pour déterminer où l'objet est stocké sur l'ordinateur de l'utilisateur. La valeur par défaut est le chemin complet du fichier SWF.

Renvoie

Aucune.

Description

Méthode qui charge toutes les données pertinentes nécessaires à la restauration de cette collection DataSet à partir d'un objet partagé. Pour enregistrer une collection DataSet dans un objet partagé, utilisez `DataSet.saveToSharedObj()`. La méthode `DataSet.loadFromSharedObject()` remplace toutes les données ou modifications en attente pouvant exister dans cette collection DataSet. Notez que le nom d'occurrence de la collection DataSet est utilisé pour identifier les données dans l'objet partagé spécifié.

Cette méthode émet une exception `DataSetError` si l'objet partagé spécifié ne peut pas être trouvé ou si un problème survient lors de la récupération des données qu'il contient.

Exemple

L'exemple suivant tente de charger un objet partagé nommé `webapp/customerInfo` et associé au jeu de données `my_ds`. La méthode est appelée dans un bloc de code `try...catch`.

```
import mx.data.components.datasetclasses.DataSetError;
try {
  my_ds.loadFromSharedObj("webapp/customerInfo");
} catch(e:DataSetError) {
  trace("Unable to load shared object.");
}
```

Voir aussi

`DataSet.saveToSharedObj()`

DataSet.locateById()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`dataSetInstance``.locateById(`*`id`*`)`

Paramètres

`id` Identifiant de chaîne de l'élément de la collection devant être localisé.

Renvoie

Une valeur booléenne.

Description

Méthode qui positionne l'itérateur courant sur l'élément de la collection dont l'ID correspond à *id*. Cette méthode renvoie `true` si l'ID spécifié correspond à celui d'un élément de la collection, ou `false` dans le cas contraire.

Exemple

L'exemple suivant utilise `DataSet.find()` pour rechercher un élément dans la collection actuelle dont les champs `name` et `id` contiennent respectivement les valeurs `"Bobby"` et 105. Si l'élément est trouvé, `DataSet.getItemId()` sert à obtenir son identificateur unique dans la collection, et `DataSet.locateById()` à positionner l'itérateur courant sur cet élément.

Pour tester cet exemple, faites glisser un composant DataSet sur la scène et nommez l'occurrence, `student_ds`. Ajoutez deux propriétés, `name` (String) et `id` (Number), au composant DataSet à l'aide de l'onglet Schéma de l'inspecteur de composants. Si vous ne possédez pas encore de copie du clip compilé `DataBindingClasses` dans votre bibliothèque, faites glisser une copie du clip compilé de la bibliothèque Classes (Fenêtre > Bibliothèques communes > Classes). Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
student_ds.addItem({name:"Barry", id:103});
student_ds.addItem({name:"Bobby", id:105});
student_ds.addItem({name:"Billy", id:107});

trace("Before find() > " + student_ds.currentItem.name); // Billy

var studentID:String;
student_ds.addSort("id", ["name","id"]);
if (student_ds.find(["Bobby", 105])) {
  studentID = student_ds.getItemId();
  student_ds.locateById(studentID);
  trace("After find() > " + student_ds.currentItem.name); // Bobby
} else {
  trace("We lost Billy!");
}
```

Voir aussi

`DataSet.applyUpdates()`, `DataSet.find()`, `DataSet.getItemId()`

DataSet.logChanges

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.logChanges

Description

Propriété ; valeur booléenne qui spécifie si les modifications apportées au jeu de données ou à ses éléments doivent être enregistrées (true) ou non (false) dans DataSet.deltaPacket.

Lorsque cette propriété est définie sur true, les opérations effectuées au niveau de la collection et au niveau de l'élément sont enregistrées. Les modifications apportées au niveau de la collection comprennent l'ajout et la suppression d'éléments de la collection. Les modifications apportées au niveau de l'élément comprennent les modifications de propriété apportées aux éléments et les appels de méthode effectués sur les éléments via le composant DataSet.

Exemple

L'exemple suivant désactive l'enregistrement pour l'objet DataSet appelé userData.

```
user_ds.logChanges = false;
```

Voir aussi

DataSet.deltaPacket

DataSet.modelChanged

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Description

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.modelChanged = function (eventObj:Object):Void {
  // ...
};
dataSetInstance.addEventListener("modelChanged", listenerObject);
```

Utilisation 2 :

```
on (modelChanged) {
  // ...
}
```

Description

Evénement ; diffusé lorsque la collection est modifiée de quelque manière que ce soit (par exemple, lorsque des éléments sont supprimés de la collection ou ajoutés à cette dernière, lorsque la valeur d'une propriété d'élément est modifiée ou lorsque la collection est filtrée ou triée).

L'objet événement (*eventObj*) possède les propriétés suivantes :

`target` L'objet DataSet qui a généré l'événement.

`type` Chaîne `"iteratorScrolled"`.

`firstItem` Index (numéro) du premier élément de la collection affecté par la modification.

`lastItem` Index (numéro) du dernier élément de la collection affecté par la modification (équivaut à `firstItem` si un seul élément a été affecté).

`fieldName` Chaîne qui contient le nom du champ affecté. Cette propriété est `undefined` sauf si la modification concernait une propriété de l'objet DataSet.

`eventName` Chaîne qui décrit la modification apportée. Cette valeur peut être :

Valeur de chaîne	Description
`"addItems"`	Une série d'éléments a été ajoutée.
`"filterModel"`	Le modèle a été filtré et l'affichage doit être actualisé (réinitialisez la position de défilement).
`"removeItems"`	Une série d'éléments a été supprimée.
`"schemaLoaded"`	La définition des champs du fournisseur de données a été déclarée.
`"sort"`	Les données ont été triées.
`"updateAll"`	L'ensemble de l'affichage doit être actualisé, en excluant la position de défilement.

Valeur de chaîne	Description
`"updateColumn"`	Une définition de champ complète doit être actualisée dans le fournisseur de données.
`"updateField"`	Un champ a été modifié dans un élément et doit être actualisé.
`"updateItems"`	Une série d'éléments doit être actualisée.

Exemple

Dans l'exemple suivant, l'événement modelChanged est distribué à chaque fois qu'un élément est ajouté ou supprimé du jeu de données :

```
my_ds.addEventListener("modelChanged", onModelChanged);
function onModelChanged(evt_obj:Object):Void {
  trace("[evénement =" + evt_obj.eventName + "] le jeu de données possède
  des " + evt_obj.target.items.length + " items.");
}
my_ds.addItem({name:"Apples", price:14});
my_ds.addItem({name:"Bananas", price:8});
my_ds.removeItemAt(0);
```

Dans l'exemple suivant, le bouton Supprimer un élément est désactivé si les éléments ont été supprimés de la collection et si l'objet DataSet cible de contient plus d'éléments :

```
my_ds.addEventListener("modelChanged", onModelChanged);
function onModelChanged(evt_obj:Object):Void {
  trace("model changed, DataSet now has " + evt_obj.target.items.length + "
  items");
}
// Désactivation des événements pour le jeu de données.
my_ds.disableEvents();

my_ds.addItem({name:"Apples", price:14});
my_ds.addItem({name:"Bananas", price:8});

trace("Before:");
traceItems();

my_ds.last();
while (my_ds.hasPrevious()) {
   my_ds.price *= 0.5; // Tout à 50 % !
   my_ds.previous();
}

trace("After:");
traceItems();

// Indication au jeu de données qu'il doit mettre à jour
// les contrôles maintenant.
my_ds.enableEvents();
```

```
function traceItems(label:String):Void {
  for (var i:Number = 0; i < my_ds.items.length; i++) {
    trace("\t" + my_ds.items[i].name + " - $" + my_ds.items[i].price);
  }
  trace("");
}
```

Voir aussi

DataSet.isEmpty()

DataSet.newItem

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.newItem = function (eventObj:Object) {
  // ...
};
dataSetInstance.addEventListener("newItem", listenerObject);
```

Utilisation 2 :

```
on (newItem) {
  // ...
}
```

Description

Evénement diffusé lorsqu'un nouvel objet de transfert est construit au moyen de DataSet.createItem(). Un écouteur pour cet événement peut modifier l'élément avant son ajout à la collection.

L'objet événement (*eventObj*) possède les propriétés suivantes :

`target` L'objet DataSet qui a généré l'événement.

`type` Chaîne `"iteratorScrolled"`.

`item` Référence à l'élément créé.

Exemple

L'exemple suivant modifie un élément nouvellement créé avant son ajout à la collection :

```
function newItemEvent(evt_obj:Object):Void {
  var employee:Object = evt_obj.item;
  employee.name = "newGuy";
  // Les données de propriété sont au format XML.
  employee.zip =
  employee.getPropertyData().firstChild.childNodes[1].attributes.zip;
}
employees_ds.addEventListener("newItem", newItemEvent);
```

DataSet.next()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.next()

Renvoie

Aucune.

Description

Méthode qui transforme l'élément suivant de la vue actuelle de la collection en élément actif. Les éléments qui se trouvent dans la vue actuelle dépendent des paramètres de filtre et d'étendue courants.

Exemple

L'exemple suivant se répète sur tous les éléments de la vue actuelle de la collection (en commençant par le début), puis calcule la propriété `price` de chaque élément :

```
my_ds.addItem({name:"item a", price:16});
my_ds.addItem({name:"item b", price:9});

my_ds.first();
while (my_ds.hasNext()) {
  my_ds.currentItem.price *= 0.5; // Tout à 50 % !
  my_ds.next();
}

for (var i in my_ds.items) {
  trace(my_ds.items[i].name + ": " + my_ds.items[i].price);
}
```

Voir aussi

DataSet.first(), DataSet.hasNext()

DataSet.previous()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.previous()

Renvoie

Aucune.

Description

Méthode qui transforme l'élément précédent de la vue actuelle de la collection en élément actif. Les éléments qui se trouvent dans la vue actuelle dépendent des paramètres de filtre et d'étendue courants.

Exemple

L'exemple suivant lit en boucle chaque élément d'un jeu de données et effectue le suivi des prix de chaque élément :

```
my_ds.addItem({name:"item a", price:16});
my_ds.addItem({name:"item b", price:9});
my_ds.last();
while (my_ds.hasPrevious()) {
  trace(my_ds.currentItem.price);
  my_ds.previous();
}
```

L'exemple suivant boucle sur tous les éléments de l'affichage actuel de la collection, en commençant par le dernier, et effectue un calcul sur un champ dans chacun d'eux :

```
my_ds.last();
while (my_ds.hasPrevious()) {
  my_ds.price *= 0.5; // Tout à 50 % !
  my_ds.previous();
}
```

Voir aussi

DataSet.first(), DataSet.hasNext()

DataSet.properties

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.properties

Description

Propriété (lecture seule) qui renvoie un objet contenant toutes les propriétés (champs) exposées pour tout objet de transfert dans cette collection.

Exemple

L'exemple suivant affiche tous les noms des propriétés de l'objet DataSet appelé `my_ds`.

```
var i:String;
for (i in my_ds.properties) {
  trace("field '" + i + "' has value " + my_ds.properties[i]);
}
```

DataSet.readOnly

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.readOnly

Description

Propriété ; valeur booléenne spécifiant si cette collection peut être modifiée (`false`) ou si elle est en lecture seule (`true`). La définition de cette propriété sur `true` empêche les mises à jour de la collection. La valeur par défaut est `false`.

Vous pouvez également définir cette propriété dans l'inspecteur des propriétés.

Exemple

L'exemple suivant définit l'objet DataSet appelé `my_ds` en lecture seule, puis tente de modifier la valeur d'une propriété appartenant à l'élément actif dans la collection. Ceci génère une exception `DataSetError`.

```
import mx.data.components.datasetclasses.DataSetError;
my_ds.readOnly = true;
try {
  // Cette opération génère une exception.
  my_ds.addItem({name:'Joe'});
} catch (e:DataSetError) {
  // Le tri spécifié 'name' n'existe pas pour DataSet 'my_ds'.
  trace("DataSetError >> " + e.message);
}
```

Voir aussi

`DataSet.currentItem`

DataSet.removeAll()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`dataSetInstance``.removeAll()`

Valeur renvoyée

Aucune.

Description

Méthode qui supprime tous les éléments de la collection DataSet.

Exemple

L'exemple suivant supprime tous les éléments de la collection DataSet `contact_ds` :

```
contact_ds.removeAll();
```

DataSet.removeItem

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.removeItem = function (eventObj:Object):Void {
  // ...
};
dataSetInstance.addEventListener("removeItem", listenerObject);
```

Utilisation 2 :

```
on (removeItem) {
  // ...
}
```

Description

Evénement généré juste avant la suppression d'un nouvel élément de cette collection.

Si vous définissez la propriété `result` de l'objet événement sur `false`, la suppression est annulée. Si vous la définissez sur `true`, elle est autorisée.

L'objet événement (*eventObj*) possède les propriétés suivantes :

`target` L'objet DataSet qui a généré l'événement.

`type` Chaîne`"removeItem"`.

`item` Référence à l'élément de la collection à supprimer.

`result` Valeur booléenne indiquant si l'élément doit être supprimé. Par défaut, cette valeur est `true`.

Exemple

Dans l'exemple suivant, un gestionnaire d'événements `on(removeItem)` annule la suppression du nouvel élément si une fonction définie par l'utilisateur, appelée `userHasAdminPrivs()`, renvoie `false`. Dans le cas contraire, la suppression est autorisée :

```
on (removeItem) {
  if (globalObj.userHasAdminPrivs()) {
    // Autorise la suppression de l'élément.
    eventObj.result = true;
  } else {
    // Ne pas autoriser la suppression de l'élément ;
    // l'utilisateur ne dispose pas des droits d'administrateur.
    eventObj.result = false;
  }
}
```

Le gestionnaire d'événements `removeItem` suivant annule la suppression de l'élément existant si une fonction définie par l'utilisateur, appelée `userHasAdminPrivs()`, renvoie la valeur `false`. Dans le cas contraire, la suppression de l'élément est autorisée :

```
function userHasAdminPrivs():Boolean {
  return false; // Modification sur true pour autoriser les insertions.
}
function removeItemListener(evt_obj:Object):Void {
  if (userHasAdminPrivs()) {
    // Allow the item removal.
    evt_obj.result = true;
    trace("Item removed");
  } else {
    // Ne pas autoriser la suppression de l'élément.
    // L'utilisateur ne dispose pas des droits d'administrateur.
    evt_obj.result = false;
    trace("Erreur, autorisations insuffisantes");
  }
}
my_ds.addEventListener("removeItem", removeItemListener);
my_ds.addItem({name:"item a", price:16});
my_ds.addItem({name:"item b", price:9});
my_ds.removeItemAt(0);
```

Voir aussi

`DataSet.addItem`

DataSet.removeItem()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.removeItem([*item*])

Paramètres

item Elément à supprimer. Ce paramètre est facultatif.

Renvoie

Une valeur booléenne. Renvoie `true` si l'élément a été supprimé avec succès, `false` dans le cas contraire.

Description

Méthode ; supprime l'élément spécifié de la collection ou supprime l'élément actif si le paramètre *item* est omis. Cette opération est enregistrée dans `DataSet.deltaPacket` si la valeur de `DataSet.logChanges` est définie `true`.

Exemple

Le code suivant supprime l'élément à la position de l'itérateur actuelle. Pour tester cet exemple, ajoutez un composant DataSet sur la scène et nommez l'occurrence, `my_ds`. Ajoutez le code suivant à l'image 1 du scénario principal :

```
my_ds.addItem({name:"Milton", years:3});
my_ds.addItem({name:"Mark", years:3});
my_ds.addItem({name:"Sarah", years:1});
my_ds.addItem({name:"Michael", years:2});
my_ds.addItem({name:"Frank", years:2});

trace(my_ds.getLength()); // 5
trace(my_ds.currentItem.name); // Frank
my_ds.removeItem();
trace(my_ds.getLength()); // 4
trace(my_ds.currentItem.name); // Michael
```

Voir aussi

`DataSet.deltaPacket`, `DataSet.logChanges`

DataSet.removeItemAt()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.removeItemAt(*index*)

Paramètres

index Nombre supérieur ou égal à 0 correspondant à l'index de l'élément à supprimer.

Valeur renvoyée

Une valeur booléenne indiquant si l'élément a été supprimé.

Description

Méthode qui supprime l'élément à l'emplacement d'index spécifié. Un index disparaît parmi les index situés après l'index supprimé.

Cette méthode déclenche l'événement `modelChanged` avec le nom d'événement `removeItems`.

Elle déclenche en outre l'événement DataSet.removeItem, qui contient les propriétés `result` et `item`. La propriété `result` sert à déterminer si l'élément (référencé par la propriété `item` de l'événement) peut être supprimé. Par défaut, la propriété `result` est définie sur `true`. Si aucun écouteur d'événement n'est spécifié pour l'événement `removeItem`, l'élément est supprimé par défaut.

Un écouteur d'événement peut interrompre la suppression de l'élément en écoutant l'événement `removeItem` et en définissant sa propriété `result` sur `false`, comme dans l'exemple suivant :

```
function removeItem(evt_obj:Object):Void {
  // Personne n'est autorisé à supprimer l'élément
  // pour lequel customerId == 0.
  evt_obj.result = (evt_obj.item.customerId != 0);
}
```

Exemple

L'exemple suivant supprime un élément situé en première position dans le jeu de données :

```
my_ds.addItem({name:"Milton", years:3});
my_ds.addItem({name:"Mark", years:3});
my_ds.addItem({name:"Sarah", years:1});
my_ds.addItem({name:"Michael", years:2});
my_ds.addItem({name:"Frank", years:2});

trace(my_ds.getLength()); // 5
trace(my_ds.currentItem.name); // Frank
my_ds.removeItemAt(0);
trace(my_ds.getLength()); // 4
trace(my_ds.currentItem.name); // Frank
```

DataSet.removeRange()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.removeRange()

Renvoie

Aucune.

Description

Méthode : supprime les paramètres de point de fin spécifiés au moyen de `DataSet.setRange()` pour l'itérateur actuel.

Exemple

```
my_ds.addSort("name_id", ["name", "id"]);
my_ds.setRange(["Bobby", 105],["Cathy", 110]);
while (my_ds.hasNext()) {
  my_ds.gradeLevel ="5"; // Modification de tous les chiffres
                         // de cette étendue.
  my_ds.next();
}
my_ds.removeRange();
my_ds.removeSort("name_id");
```

Voir aussi

`DataSet.applyUpdates()`, `DataSet.hasNext()`, `DataSet.next()`, `DataSet.removeSort()`, `DataSet.setRange()`

DataSet.removeSort()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.removeSort(*sortName*)

Paramètres

sortName Chaîne spécifiant le nom du tri à supprimer.

Renvoie

Aucune.

Description

Méthode qui supprime le tri spécifié de cet objet DataSet, si le tri existe. Si le tri spécifié n'existe pas, cette méthode émet une exception `DataSetError`.

Exemple

L'exemple suivant crée une étendue d'éléments dans le composant DataSet et modifie la propriété gradeLevel de chaque élément. Pour tester cet exemple, faites glisser un composant DataSet sur la scène et nommez l'occurrence, `my_ds`. Le composant DataSet étant sélectionné, créez trois propriétés dans le schéma du composant DataSet à l'aide de l'onglet Schéma de l'inspecteur de composants. Nommez-les `name`, `id` et `gradeLevel` et donnez-leur respectivement les types de données String, Number et Number. Ajoutez une copie du clip compilé `DataBindingClasses` de la bibliothèque commune Classes (Fenêtre > Bibliothèques communes > Classes) et ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
my_ds.addItem({name:"Billy", id:104, gradeLevel:4});
my_ds.addItem({name:"Bobby", id:105, gradeLevel:4});
my_ds.addItem({name:"Carrie", id:106, gradeLevel:4});
my_ds.addItem({name:"Cathy", id:110, gradeLevel:4});
my_ds.addItem({name:"Mally", id:112, gradeLevel:3});

my_ds.addSort("name_id", ["name", "id"]);
my_ds.setRange(["Bobby", 105], ["Cathy", 110]);
while (my_ds.hasNext()) {
  my_ds.gradeLevel = "5"; // Modification de tous les chiffres
                          // de cette étendue.
  my_ds.next();
}
```

```
my_ds.removeRange();
my_ds.removeSort("name_id");

for (var i=0; i<my_ds.length; i++) {
  trace(my_ds.items[i].name + " > " + my_ds.items[i].gradeLevel);
}
```

Voir aussi

DataSet.applyUpdates(), DataSet.hasNext(), DataSet.next(), DataSet.removeRange(), DataSet.setRange()

DataSet.resolveDelta

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.resolveDelta = function (eventObj:Object):Void {
  // ...
};
dataSetInstance.addEventListener("resolveDelta", listenerObject);
```

Utilisation 2 :

```
on (resolveDelta) {
  // ...
}
```

Description

Evénement ; diffusé lorsque DataSet.deltaPacket se voit affecter un paquet delta dont l'ID de transaction correspond à celui d'un paquet delta, récupéré précédemment dans l'objet DataSet, et présentant des messages associés à des objets Delta ou DeltaItem dans ce paquet delta.

Cet événement vous donne la possibilité de réparer toute erreur renvoyée par le serveur lors de la tentative d'application des modifications soumises précédemment. En général, vous utilisez cet événement pour afficher une « boîte de dialogue de reconstitution » avec les valeurs en conflit, autorisant l'utilisateur à apporter les modifications appropriées aux données de sorte qu'elles puissent à nouveau être envoyées.

L'objet événement (*eventObj*) possède les propriétés suivantes :

`target` L'objet DataSet qui a généré l'événement.

`type` Chaîne `"resolveDelta"`.

`data` Tableau d'objets Delta et DeltaItem associés qui possèdent des messages dont la longueur est différente de zéro.

Exemple

L'exemple suivant affiche un formulaire appelé `reconcileForm` (masqué) et appelle une méthode sur cet objet formulaire (`setReconcileData()`) qui autorise l'utilisateur à reconstituer toute valeur conflictuelle renvoyée par le serveur :

```
import mx.data.components.datasetclasses.*;
my_ds.addEventListener("resolveDelta", onResolveDelta);
function onResolveDelta(eventObj:Object) {
  reconcileForm.visible = true;
  reconcileForm.setReconcileData(eventObj.data);
}
// Dans le code reconcileForm.
function setReconcileData(data:Array):Void {
  var di:DeltaItem;
  var ops:Array = ["property", "method"];
  var cl:Array;
  // change list
  var msg:String;
  for (var i = 0; i<data.length; i++) {
    cl = data[i].getChangeList();
    for (var j = 0; j<cl.length; j++) {
      di = cl[j];
      msg = di.message;
      if (msg.length>0) {
        trace("Le problème suivant s'est produit '"+msg+"' lors de la
  '"+ops[di.kind]+"' modification '"+di.name+"' d'une valeur de serveur
  actuelle ["+di.curValue+"], valeur envoyée ["+di.newValue+"] Veuillez le
  corriger !");
      }
    }
  }
}
```

DataSet.saveToSharedObj()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
dataSetInstance.saveToSharedObj(objName, [localPath])
```

Paramètres

objName Chaîne spécifiant le nom de l'objet partagé à créer. Le nom peut inclure des barres obliques (par exemple, « travail/adresses »). Les espaces et les caractères suivants ne sont pas autorisés dans le nom spécifié :

```
~ % & \ ; : " ' , < > ? #
```

localPath Paramètre de chaîne facultatif qui précise le chemin complet ou partiel du fichier SWF qui a créé l'objet partagé. La chaîne est utilisée pour déterminer l'endroit où l'objet est stocké sur l'ordinateur de l'utilisateur. La valeur par défaut est le chemin complet du fichier SWF.

Renvoie

Aucune.

Description

Méthode qui enregistre toutes les données pertinentes nécessaires à la restauration de cette collection DataSet dans un objet partagé. Ceci permet aux utilisateurs de travailler lorsqu'ils sont déconnectés des données source, s'il s'agit d'une ressource réseau. Cette méthode remplace toute donnée existant éventuellement dans l'objet partagé spécifié pour cette collection DataSet. Pour restaurer une collection DataSet à partir d'un objet partagé, utilisez `DataSet.loadFromSharedObj()`. Notez que le nom d'occurrence de la collection DataSet est utilisé pour identifier les données dans l'objet partagé spécifié.

Si l'objet partagé ne peut pas être créé ou qu'un problème survient lors de la purge des données vers cet objet, cette méthode émet une exception `DataSetError`.

Exemple

L'exemple suivant appelle `saveToSharedObj()` dans un bloc `try..catch` et affiche un message d'erreur si un problème survient lors de l'enregistrement des données dans l'objet partagé.

```
import mx.data.components.datasetclasses.DataSetError;
try {
  my_ds.saveToSharedObj("webapp/customerInfo");
} catch(e:DataSetError) {
  trace("Unable to create shared object");
}
```

Voir aussi

`DataSet.loadFromSharedObj()`

DataSet.schema

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`dataSetInstance``.schema`

Description

Propriété qui fournit la représentation XML du schéma pour cet objet DataSet. Le code XML affecté à cette propriété doit avoir le format suivant :

```
<?xml version="1.0"?>
<properties>
  <property name="propertyName">
    <type name="dataType" />
    <encoder name="dataType">
      <options>
        <dataFormat>format options</dataFormat>
      </options>
    </encoder>
    <kind name="dataKind">
    </kind>
  </property>
  <property> ... </property>
  ...
</properties>
```

Une exception `DataSetError` est émise si le code XML spécifié ne respecte pas le format ci-dessus.

Exemple

L'exemple suivant définit le schéma du jeu de données `my_ds` sur un nouvel objet XML contenant le code XML approprié :

```
my_ds.schema = new XML("<properties><property name="billable"> ..etc.. </
  properties>");
```

DataSet.selectedIndex

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`dataSetInstance``.selectedIndex`

Description

Propriété qui spécifie l'index sélectionné dans la collection. Vous pouvez lier cette propriété à l'élément sélectionné dans un composant DataGrid ou List, et inversement. Pour obtenir un exemple complet de cette fonction, reportez-vous à « Création d'une application avec le composant DataSet », à la page 371.

Exemple

L'exemple suivant définit l'index sélectionné d'un objet DataSet (`user_ds`) sur l'index sélectionné dans un composant DataGrid (`user_dg`).

```
user_ds.selectedIndex = user_dg.selectedIndex;
```

DataSet.setIterator()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.setIterator(*iterator*)

Paramètres

iterator Objet itérateur renvoyé par un appel à DataSet.getIterator().

Renvoie

Aucune.

Description

Méthode qui affecte l'itérateur spécifié à cet objet DataSet et le transforme en itérateur courant. L'itérateur spécifié doit venir d'un appel précédent à DataSet.getIterator() sur l'objet DataSet auquel il est affecté, sinon une exception DataSetError est émise.

Exemple

```
import mx.data.to.ValueListIterator;
myIterator:ValueListIterator = my_ds.getIterator();
myIterator.sortOn(["name"]);
my_ds.setIterator(myIterator);
```

Voir aussi

DataSet.getIterator()

DataSet.setRange()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

dataSetInstance.setRange(*startValues*, *endValues*)

Paramètres

startValues Tableau des principales valeurs des propriétés du premier objet de transfert de l'étendue.

endValues Tableau des principales valeurs des propriétés du dernier objet de transfert de l'étendue.

Renvoie

Aucune.

Description

Méthode qui définit les points de fin de l'itérateur courant. Ces points de fin définissent l'étendue dans laquelle l'itérateur agit. Ceci n'est valable que si un tri valide a été défini pour l'itérateur actuel au moyen de `DataSet.addSort()`.

Si vous souhaitez regrouper les valeurs, il est plus efficace de définir une étendue pour l'itérateur actuel que d'utiliser une fonction de filtre (voir `DataSet.filterFunc`).

Exemple

L'exemple suivant sélectionne une étendue d'étudiants et suit chacun de leurs noms dans le panneau Sortie :

```
my_ds.addItem({name:"Billy", id:104, gradeLevel:4});
my_ds.addItem({name:"Bobby", id:105, gradeLevel:4});
my_ds.addItem({name:"Carrie", id:106, gradeLevel:4});
my_ds.addItem({name:"Cathy", id:110, gradeLevel:4});
my_ds.addItem({name:"Mally", id:112, gradeLevel:3});
my_ds.addSort("name_id",["name", "id"]);
my_ds.setRange(["Bobby", 105],["Cathy", 110]);
while (my_ds.hasNext()) {
  trace(my_ds.name); // Bobby..Cathy
  my_ds.next();
}
```

Voir aussi

`DataSet.addSort()`, `DataSet.hasNext()`, `DataSet.next()`, `DataSet.removeRange()`, `DataSet.removeSort()`

DataSet.skip()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`dataSetInstance``.skip(`*`offSet`*`)`

Paramètres

offSet Nombre entier spécifiant le nombre d'enregistrements en fonction duquel déplacer la position de l'itérateur.

Renvoie

Aucune.

Description

Méthode ; avance ou recule la position de l'itérateur dans la collection selon le nombre spécifié par *offSet*. Les valeurs de *offSet* positives avancent la position de l'itérateur, les valeurs négatives la font reculer.

Si le décalage spécifié se situe au-delà du début (ou de la fin) de la collection, l'itérateur se place au début (ou à la fin) de la collection.

Exemple

L'exemple suivant positionne l'itérateur actuel sur le premier élément de la collection, puis se déplace vers l'avant-dernier élément, et effectue un calcul sur un champ appartenant à cet élément :

```
my_ds.addItem({name:"Billy", id:104, gradeLevel:4});
my_ds.addItem({name:"Carrie", id:106, gradeLevel:4});
my_ds.addItem({name:"Mally", id:112, gradeLevel:3});
my_ds.addItem({name:"Cathy", id:110, gradeLevel:4});
my_ds.addItem({name:"Bobby", id:105, gradeLevel:4});
my_ds.first();
var itemsToSkip:Number = 3;
trace(my_ds.currentItem.name); // Billy
my_ds.skip(itemsToSkip);
trace(my_ds.currentItem.name); // Mally
```

DataSet.useSort()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
dataSetInstance.useSort(sortName, order)
```

Paramètres

sortName Chaîne qui contient le nom du tri à utiliser.

order Valeur entière qui indique l'ordre de tri. La valeur doit être `DataSetIterator.Ascending` ou `DataSetIterator.Descending`.

Renvoie

Aucune.

Description

Méthode qui définit le tri de l'itérateur courant sur le tri spécifié par *sortName*, lorsque celui-ci existe. Si le tri spécifié n'existe pas, cette méthode émet une exception `DataSetError`.

Pour créer un tri, utilisez `DataSet.addSort()`.

Exemple

L'exemple suivant utilise `DataSet.hasSort()` pour vérifier l'existence d'un tri appelé `"customer"`. S'il existe, le code appelle `DataSet.useSort()` pour faire de `"customer"` le tri actuel. Sinon, le code crée un tri par ce nom en utilisant `DataSet.addSort()`.

```
if (my_ds.hasSort("customer")) {
  my_ds.useSort("customer");
} else {
  my_ds.addSort("customer", ["customer"], DataSetIterator.Descending);
}
```

Voir aussi

`DataSet.applyUpdates()`, `DataSet.hasSort()`

CHAPITRE 15

Composant DateField

15

Ce composant est un champ de texte non sélectionnable qui affiche la date accompagnée d'une icône de calendrier, sur sa droite.

REMARQUE

Le composant DateField est pris en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Si aucune date n'a été sélectionnée, le champ de texte est vide et la date du mois en cours s'affiche dans le sélecteur de dates. Lorsqu'un utilisateur clique dans le cadre de sélection de ce champ de date, un sélecteur présentant les dates du mois de la date sélectionnée apparaît. Lorsque le sélecteur de dates est ouvert, les utilisateurs peuvent utiliser les boutons de défilement des mois pour faire défiler les mois et les années, afin de sélectionner une date. Lorsque vous sélectionnez une date, le sélecteur de dates se ferme et la sélection est entrée dans le champ de date.

CONSEIL

Le champ de date est effacé lorsqu'un utilisateur sélectionne deux fois la même date. Pour empêcher les utilisateurs de désélectionner par inadvertance leur date souhaitée, reportez-vous à l'exemple relatif à « DateField.selectedDate » à la page 474.

L'aperçu en direct du composant DateField ne reflète pas les valeurs indiquées par l'inspecteur Propriétés ou des composants au cours de la programmation car il s'agit d'un composant contextuel non visible lors de la programmation.

Utilisation du composant DateField

Le composant DateField peut être utilisé là où vous souhaitez qu'un utilisateur sélectionne une date. Par exemple, dans un système de réservation de chambres d'hôtel qui comprend des dates sélectionnables et des dates désactivées. Vous pouvez également l'utiliser dans une application qui affiche des événements en cours, spectacles ou réunions, lorsque l'utilisateur sélectionne une date.

Paramètres du composant DateField

Vous pouvez définir les paramètres de programmation suivants pour chaque composant DateField dans l'inspecteur Propriétés ou l'inspecteur des composants :

dayNames définit les noms des jours de la semaine. La valeur est un tableau et la valeur par défaut est `["S", "M", "T", "W", "T", "F", "S"]`.

disabledDays indique les jours désactivés de la semaine. Ce paramètre est un tableau qui peut comprendre un maximum de sept valeurs. La valeur par défaut est `[]` (tableau vide).

firstDayOfWeek indique le jour de la semaine (0-6, 0 étant le premier élément du tableau `dayNames`) affiché dans la première colonne du sélecteur de dates. Cette propriété modifie l'ordre d'affichage des colonnes des jours.

La valeur par défaut est 0, qui correspond à « S » pour sunday.

monthNames définit les noms des mois affichés dans la ligne d'en-tête du calendrier. La valeur est un tableau et la valeur par défaut est `["January", "February", "March", "April", "May", "June", "July", "August", "September", "October","November", "December"]`.

showToday indique si la date du jour doit être mise en surbrillance. La valeur par défaut est `true`.

Vous pouvez rédiger du code ActionScript pour contrôler ces options et d'autres options du composant DateField à l'aide des propriétés, méthodes et événements ActionScript. Pour plus d'informations, voir « Classe DateField », à la page 454.

Création d'une application avec le composant DateField

La procédure suivante explique comment ajouter un composant DateField à une application lors de la programmation. Dans cet exemple, le composant DateField autorise l'utilisateur à choisir une date dans un système de réservation de vols. Toutes les dates qui précèdent la date du jour doivent être désactivées. Une plage de 15 jours du mois de décembre doit également être désactivée pour créer une période de vacances. Certains vols ne sont pas disponibles les lundis. Tous les lundis doivent donc être désactivés pour ces vols.

Pour créer une application avec le composant DateField :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Double-cliquez sur le composant DateField dans le panneau Composants pour l'ajouter sur la scène.
3. Dans l'inspecteur Propriétés, entrez le nom d'occurrence **flightCalendar**.
4. Dans le panneau Actions, entrez le code suivant dans l'image 1 du scénario pour définir la plage des dates sélectionnables :

   ```
   flightCalendar.selectableRange = {rangeStart:new Date(2001, 9, 1),
     rangeEnd:new Date(2003, 11, 1)};
   ```

   Ce code affecte une valeur à la propriété `selectableRange` dans un objet ActionScript qui contient deux objets Date dont les variables sont nommées `rangeStart` et `rangeEnd`. Il définit les limites inférieure et supérieure de la plage dans laquelle l'utilisateur peut choisir une date.
5. Dans le panneau Actions, entrez le code suivant dans l'image 1 du scénario pour définir les plages de dates désactivées, une en décembre et une pour toutes les dates précédant la date du jour :

   ```
   flightCalendar.disabledRanges = [{rangeStart: new Date(2003, 11, 15),
     rangeEnd: new Date(2003, 11, 31)}, {rangeEnd: new Date(2003, 6, 16)}];
   ```
6. Dans le panneau Actions, entrez le code suivant dans l'image 1 du scénario pour désactiver les lundis :

   ```
   flightCalendar.disabledDays=[1];
   ```
7. Choisissez Contrôle > Tester l'animation.

Pour créer une occurrence du composant DateField à l'aide d'ActionScript :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser le composant DateField du panneau Composants jusqu'à la bibliothèque du document actif.

   Cette opération permet d'ajouter le composant à la bibliothèque, mais elle ne permet pas de le rendre visible dans l'application.
3. Sélectionnez la première image dans le scénario principal, ouvrez le panneau Actions et indiquez le code suivant :

   ```
   this.createClassObject(mx.controls.DateField, "my_df", 1);
   ```

   Ce script utilise la méthode `UIObject.createClassObject()` pour créer l'occurrence du composant DateField.
4. Sélectionnez Contrôle > Tester l'animation.

Personnalisation du composant DateField

Vous pouvez transformer un composant DateField horizontalement au cours de la programmation et à l'exécution. Lors de la programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez la méthode `setSize()` (voir `UIObject.setSize()`). La définition de la largeur ne modifie pas les dimensions du sélecteur de dates dans le composant DateField. Vous pouvez néanmoins utiliser la propriété `pullDown` pour accéder au composant DateChooser et définir ses dimensions.

Utilisation des styles avec le composant DateField

Vous pouvez définir les propriétés des styles pour modifier l'apparence d'une occurrence de champ de date. Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, consultez « Utilisation de styles pour personnaliser la couleur et le texte des composants » dans *Utilisation des composants ActionScript 2.0*.

Un composant DateField prend en charge les styles suivants :

Style	Thème	Description
`themeColor`	Halo	Couleur de l'éclat pour les dates survolées et sélectionnées. `"haloGreen"`, `"haloBlue"` et `"haloOrange"` sont des valeurs possibles. La valeur par défaut est `"haloGreen"`.
`backgroundColor`	Les deux	Couleur d'arrière-plan. La valeur par défaut est 0xEFEBEF (gris clair).
`borderColor`	Les deux	Couleur de bordure. La valeur par défaut est 0x919999. La liste déroulante du composant DateField utilise comme bordure une ligne unie et large d'un pixel. Cette bordure n'est pas modifiable par l'application d'enveloppes ou de styles.
`headerColor`	Les deux	Couleur d'arrière-plan de l'en-tête de la liste déroulante. La couleur par défaut est le blanc.
`rollOverColor`	Les deux	Couleur d'arrière-plan d'une date survolée. La couleur par défaut est 0xE3FFD6 (vert vif) pour le thème Halo et 0xAAAAAA (gris clair) pour le thème Sample.
`selectionColor`	Les deux	Couleur d'arrière-plan de la date sélectionnée. La couleur par défaut est 0xCDFFC1 (vert clair) pour le thème Halo et 0xEEEEEE (gris très clair) pour le thème Sample.
`todayColor`	Les deux	Couleur d'arrière-plan de la date du jour. La valeur par défaut est 0x666666 (gris foncé).
`color`	Les deux	Couleur du texte. La valeur par défaut est 0x0B333C pour le thème Halo et vide pour le thème Sample.
`disabledColor`	Les deux	Couleur du texte lorsque le composant est désactivé. La couleur par défaut est 0x848384 (gris foncé).
`embedFonts`	Les deux	Valeur booléenne qui indique si la police spécifiés dans `fontFamily` est de type intégré. Ce style doit être défini sur `true` si `fontFamily` fait référence à une police intégrée. Dans le cas contraire, la police intégrée n'est pas utilisée. Si ce style est défini sur `true` et que `fontFamily` ne fait pas référence à une police intégrée, aucun texte n'est affiché. La valeur par défaut est `false`.

Style	Thème	Description
`fontFamily`	Les deux	Nom de la police du texte. La valeur par défaut est `"_sans"`.
`fontSize`	Les deux	Taille de la police, en points. La valeur par défaut est 10.
`fontStyle`	Les deux	Style de police : `"normal"` ou `"italic"`. La valeur par défaut est `"normal"`.
`fontWeight`	Les deux	Epaisseur de la police : `"none"` ou `"bold"`. La valeur par défaut est `"none"`. Tous les composants peuvent également accepter la valeur `"normal"` au lieu de `"none"` pendant un appel à la méthode `setStyle()`, mais les appels suivants à `getStyle()` renvoient `"none"`.
`textDecoration`	Les deux	Décoration du texte : `"none"` ou `"underline"`. La valeur par défaut est `"none"`.

Le composant DateField utilise quatre catégories de texte pour afficher le nom du mois, les jours de la semaine, la date du jour et les dates ordinaires. Les propriétés de style de texte définies dans le composant DateField lui-même contrôlent le texte des dates ordinaires et le texte affiché en état réduit et fournissent également les valeurs par défaut de l'autre texte. Pour définir des styles de texte pour des catégories de texte spécifiques, servez-vous des déclarations de styles suivantes au niveau de la classe.

Nom de la déclaration	Description
`HeaderDateText`	Nom du mois.
`WeekDayStyle`	Jours de la semaine.
`TodayStyle`	Date du jour.

L'exemple suivant montre comment définir le nom du mois et les jours de la semaine en rouge foncé.

```
_global.styles.HeaderDateText.setStyle("color", 0x660000);
_global.styles.WeekDayStyle.setStyle("color", 0x660000);
```

Utilisation des enveloppes avec le composant DateField

Le composant DateField utilise des enveloppes pour représenter les états visuels de l'icône contextuelle, une occurrence RectBorder pour la bordure de la zone de saisie et une occurrence DateChooser pour le sélecteur de dates contextuel. Pour appliquer des enveloppes à l'icône contextuelle lors de la création, modifiez les symboles d'enveloppe dans le dossier Flash UI Components 2/Themes/MMDefault/DateField Assets/States dans la bibliothèque de l'un des fichiers FLA de thèmes. Pour plus d'informations, consultez « A propos de l'application d'enveloppes aux composants » dans *Utilisation des composants ActionScript 2.0*. Pour plus d'informations sur l'application d'enveloppes aux occurrences RectBorder et DateChooser, reportez-vous à « Classe RectBorder », à la page 1111 et à « Utilisation des enveloppes avec le composant DateChooser », à la page 352.

Outre les enveloppes employées par les sous-composants mentionnés ci-dessus, un composant DateField utilise les propriétés d'enveloppes suivantes pour envelopper dynamiquement son icône contextuelle :

Propriété	Description
`openDateUp`	Etat relevé de l'icône contextuelle.
`openDateDown`	Etat enfoncé de l'icône contextuelle.
`openDateOver`	Etat survolé de l'icône contextuelle.
`openDateDisabled`	Etat désactivé de l'icône contextuelle.

Pour créer des symboles de clip pour les enveloppes DateField :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Choisissez Fichier > Importer > Ouvrir une bibliothèque externe et sélectionnez le fichier HaloTheme.fla.

   Ce fichier est stocké dans le dossier Configuration au niveau de l'application. Pour connaître son emplacement exact dans votre système d'exploitation, reportez-vous à « Présentation des thèmes » dans *Utilisation des composants ActionScript 2.0*.
3. Dans le panneau Bibliothèque du thème, choisissez Flash UI Components 2/Themes/MMDefault et faites glisser le dossier DateField Assets dans la bibliothèque de votre document.
4. Développez le dossier DateField Assets dans la bibliothèque de votre document.
5. Assurez-vous que l'option Exporter dans la première image est sélectionnée pour le symbole DateFieldAssets.
6. Développez le dossier DateField Assets/States dans la bibliothèque de votre document.

7. Ouvrez les symboles que vous souhaitez personnaliser pour la modification.

   Par exemple, ouvrez le symbole openIconUp.
8. Personnalisez le symbole selon vos besoins.

   Par exemple, tracez une flèche dirigée vers le bas sur l'image du calendrier.
9. Répétez les étapes 7 et 8 pour tous les symboles devant être personnalisés.

   Par exemple, tracez une flèche dirigée vers le bas sur tous les symboles.
10. Cliquez sur le bouton Précédent pour revenir au scénario principal.
11. Faites glisser un composant DateField sur la scène.
12. Sélectionnez Contrôle > Tester l'animation.

Classe DateField

Héritage MovieClip > Classe UIObject > Classe UIComponent > ComboBase > DateField

Nom de classe ActionScript mx.controls.DateField

Les propriétés de la classe DateField vous permettent d'accéder à la date sélectionnée et au mois et à l'année affichés. Vous pouvez également définir les noms des jours et des mois, indiquer les dates désactivées et les dates sélectionnables, définir le premier jour de la semaine et indiquer si la date du jour doit être mise en surbrillance.

La définition d'une propriété de la classe DateField avec ActionScript annule le paramètre du même nom défini dans l'inspecteur Propriétés ou des composants.

Toutes les classes de composants ont une propriété `version` qui correspond à une propriété de classe. Les propriétés de classe ne sont disponibles que dans la classe elle-même. La propriété `version` renvoie une chaîne qui indique la version du composant. Pour accéder à cette propriété, utilisez le code suivant :

```
trace(mx.controls.DateField.version);
```

REMARQUE

Le code `trace(myDateFieldInstance.version);` renvoie `undefined`.

Méthodes de la classe DateField

Le tableau suivant présente les méthodes de la classe DateField.

Méthode	Description
`DateField.close()`	Ferme le sous-composant DateChooser contextuel.
`DateField.open()`	Ouvre le sous-composant DateChooser contextuel.

Méthodes héritées de la classe UIObject

Le tableau suivant énumère les méthodes de la classe DateField héritées de la classe UIObject. Pour appeler ces méthodes à partir de l'objet DateField, utilisez le formulaire *`dateFieldInstance.methodName`*.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image active.
`UIObject.setSize()`	Redimensionne l'objet à la taille demandée.
`UIObject.setSkin()`	Définit une enveloppe dans l'objet.
`UIObject.setStyle()`	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant énumère les méthodes de la classe DateField héritées de la classe UIComponent. Pour appeler ces méthodes à partir de l'objet DateField, utilisez le formulaire *dateFieldInstance.methodName*.

Méthode	Description
`UIComponent.getFocus()`	Renvoie une référence à l'objet ayant le focus.
`UIComponent.setFocus()`	Attribue le focus à l'occurrence de composant.

Propriétés de la classe DateField

Le tableau suivant présente les propriétés de la classe DateField.

Propriété	Description
`DateField.dateFormatter`	Fonction qui formate la date à afficher dans le champ de texte.
`DateField.dayNames`	Tableau indiquant les noms des jours de la semaine.
`DateField.disabledDays`	Tableau indiquant les jours désactivés de la semaine.
`DateField.disabledRanges`	Plage de dates (ou date unique) désactivées.
`DateField.displayedMonth`	Nombre indiquant l'élément du tableau `monthNames` à afficher.
`DateField.displayedYear`	Nombre indiquant l'année à afficher.
`DateField.firstDayOfWeek`	Nombre indiquant l'élément du tableau `dayNames` à afficher dans la première colonne du composant DateField.
`DateField.monthNames`	Tableau de chaînes indiquant les noms des mois.
`DateField.pullDown`	Référence au sous-composant DateChooser. Cette propriété est en lecture seule.
`DateField.selectableRange`	Plage de dates (ou date unique) sélectionnables.
`DateField.selectedDate`	Objet Date indiquant la date sélectionnée.
`DateField.showToday`	Valeur booléenne indiquant si la date du jour est mise en surbrillance.

Propriétés héritées de la classe UIObject

Le tableau suivant énumère les propriétés de la classe DateField héritées de la classe UIObject. Pour accéder à ces propriétés à partir de l'objet DateField, utilisez le formulaire *dateFieldInstance.propertyName*.

Propriété	Description
UIObject.bottom	Position du bord inférieur de l'objet par rapport au bord inférieur de son parent. Lecture seule.
UIObject.height	Hauteur de l'objet, en pixels. Lecture seule.
UIObject.left	Bord gauche de l'objet, en pixels. Lecture seule.
UIObject.right	Position du bord droit de l'objet par rapport au bord droit de son parent. Lecture seule.
UIObject.scaleX	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
UIObject.scaleY	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
UIObject.top	Position du bord supérieur de l'objet par rapport à son parent. Lecture seule.
UIObject.visible	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
UIObject.width	Largeur de l'objet, en pixels. Lecture seule.
UIObject.x	Bord gauche de l'objet, en pixels. Lecture seule.
UIObject.y	Bord supérieur de l'objet, en pixels. Lecture seule.

Propriétés héritées de la classe UIComponent

Le tableau suivant énumère les propriétés de la classe DateField héritées de la classe UIComponent. Pour accéder à ces propriétés à partir de l'objet DateField, utilisez le formulaire *dateFieldInstance.propertyName*.

Propriété	Description
UIComponent.enabled	Indique si le composant peut recevoir le focus et la saisie.
UIComponent.tabIndex	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Evénements de la classe DateField

Le tableau suivant présente les événements de la classe DateField.

Evénement	Description
`DateField.change`	Diffusé lorsqu'une date est sélectionnée.
`DateField.close`	Diffusé lors de la fermeture du sous-composant DateChooser.
`DateField.open`	Diffusé lors de l'ouverture du sous-composant DateChooser.
`DateField.scroll`	Diffusé lorsque l'utilisateur clique sur les boutons des mois.

Evénements hérités de la classe UIObject

Le tableau suivant énumère les événements de la classe DateField hérités de la classe UIObject.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant énumère les événements de la classe DateField hérités de la classe UIComponent.

Evénement	Description
`UIComponent.focusIn`	Diffusé lorsqu'un objet reçoit le focus.
`UIComponent.focusOut`	Diffusé lorsqu'un objet perd le focus.
`UIComponent.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIComponent.keyUp`	Diffusé lorsqu'une touche est relâchée.

DateField.change

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.change = function(eventObject:Object) {
  // ...
};
dateFieldInstance.addEventListener("change", listenerObject);
```

Utilisation 2 :

```
on (change) {
  // ...
}
```

Description

Evénement diffusé à tous les écouteurs enregistrés lorsqu'une date est sélectionnée.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence du composant (*dateFieldInstance*) distribue un événement (ici, `change`) qui est géré par une fonction (également appelée *gestionnaire*) dans un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée au déclenchement de l'événement. Lorsqu'il est déclenché, l'événement transmet automatiquement un objet événement (*eventObject*) à la méthode de l'objet écouteur. Chaque objet événement est doté de propriétés contenant des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence DateField. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de composant. Par exemple, le code suivant, associé au champ de date `my_df`, envoie « _level0.my_df » vers le panneau Sortie :

```
on (change) {
  trace(this);
}
```

Exemple

L'exemple suivant, écrit sur une image du scénario, envoie un message au panneau Sortie lorsqu'un champ de date nommé `my_df` est modifié. La première ligne de code crée un objet écouteur nommé `dfListener`. La deuxième ligne définit une fonction pour l'événement `change` sur l'objet écouteur. La fonction comporte une instruction `trace()` qui utilise l'objet événement automatiquement transmis à cette fonction (ici `evt_obj`) pour générer un message. La propriété `target` d'un objet événement est le composant qui a généré l'événement (dans cet exemple, `my_df`). L'utilisateur accède à la propriété `DateField.selectedDate` à partir de la propriété `target` de l'objet événement. La dernière ligne appelle la méthode `EventDispatcher.addEventListener()` à partir de `my_df` et lui transmet l'événement `change` et l'objet écouteur `dfListener` comme paramètres.

```
// Création d'un objet écouteur.
var dfListener:Object = new Object();
dfListener.change = function(evt_obj:Object){
 var thisDate:Date = evt_obj.target.selectedDate;
    trace("date selected: " + thisDate);
}

// Ajout d'un objet écouteur au composant DateField.
my_df.addEventListener("change", dfListener);
```

DateField.close()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dateFieldInstance.close()

Valeur renvoyée

Aucune.

Description

Méthode qui ferme le menu contextuel.

Exemple

Le code suivant ferme le composant DateChooser contextuel de l'occurrence DateField `my_df` lorsque vous cliquez sur le bouton `my_btn` :

```
// Création d'un objet écouteur.
var btnListener:Object = new Object();
btnListener.click = function() {
   my_df.close();
};

// Ajout de l'écouteur du composant Button.
my_btn.addEventListener("click", btnListener);
```

DateField.close

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.close = function(eventObject:Object) {
  // ...
};
dateFieldInstance.addEventListener("close", listenerObject);
```

Utilisation 2 :

```
on (close) {
  // ...
}
```

Description

Evénement diffusé à tous les écouteurs enregistrés lorsque le sous-composant DateChooser se ferme quand l'utilisateur clique hors de l'icône ou choisit une date.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence du composant (*dateFieldInstance*) distribue un événement (ici, close) qui est géré par une fonction (également appelée *gestionnaire*) dans un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée au déclenchement de l'événement. Lorsqu'il est déclenché, l'événement transmet automatiquement un objet événement (*eventObject*) à la méthode de l'objet écouteur. Chaque objet événement est doté de propriétés contenant des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode EventDispatcher.addEventListener() sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel au gestionnaire on() et doit être associé directement à une occurrence DateField. Le mot-clé this, utilisé dans un gestionnaire on() lié à un composant, correspond à l'occurrence de composant. Par exemple, le code suivant, associé au champ de date my_df, envoie « _level0.my_df » vers le panneau Sortie :

```
on (close) {
  trace(this);
}
```

Exemple

L'exemple suivant, écrit sur une image du scénario, envoie un message au panneau Sortie lors de la fermeture du sélecteur de dates dans my_df. La première ligne de code crée un objet écouteur intitulé dfListener. La deuxième ligne définit une fonction pour l'événement close sur l'objet écouteur. La fonction comporte une instruction trace() qui utilise l'objet événement automatiquement transmis à cette fonction (ici evt_obj) pour générer un message. La propriété target d'un objet événement est le composant qui a généré l'événement (dans cet exemple, my_df). L'utilisateur accède à la propriété selectedDate à partir de la propriété target de l'objet événement. La dernière ligne appelle la méthode EventDispatcher.addEventListener() à partir de my_df et lui transmet l'événement close et l'objet écouteur dfListener comme paramètres.

```
// Création d'un objet écouteur.
var dfListener:Object = new Object();
dfListener.close = function(evt_obj:Object){
  trace("PullDown Closed" + evt_obj.target.selectedDate);
}
// Ajout d'un objet écouteur au composant DateField.
my_df.addEventListener("close", dfListener);
```

DateField.dateFormatter

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dateFieldInstance.dateFormatter

Description

Propriété, fonction qui formate la date à afficher dans le champ de texte. La fonction doit recevoir un objet Date comme paramètre et renvoyer une chaîne au format à afficher.

Exemple

L'exemple suivant définit la fonction pour renvoyer le format de la date à afficher :

```
my_df.dateFormatter = function(d:Date){
  return d.getFullYear()+"/ "+(d.getMonth()+1)+"/ "+d.getDate();
};
```

DateField.dayNames

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dateFieldInstance.dayNames

Description

Propriété, tableau contenant les noms des jours de la semaine. Dimanche correspond au premier jour (à l'emplacement d'index 0) et le reste des jours suit, dans l'ordre. La valeur par défaut est `["S", "M", "T", "W", "T", "F", "S"]`.

Exemple

L'exemple suivant change la valeur du cinquième jour de la semaine (jeudi) en la faisant passer de « T » à « R » :

```
my_df.dayNames[4] = "R";
```

L'exemple suivant modifie la valeur de tous les jours, en conséquence :

```
my_df.dayNames = new Array("Su", "Mo", "Tu", "We", "Th", "Fr", "Sa");
```

DateField.disabledDays

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dateFieldInstance.disabledDays

Description

Propriété, tableau indiquant les jours désactivés de la semaine. Toutes les dates du mois qui correspondent au jour spécifié sont désactivées. Les éléments de ce tableau peuvent avoir des valeurs comprises entre 0 (dimanche) et 6 (samedi). La valeur par défaut est `[]` (tableau vide).

Exemple

L'exemple suivant désactive les dimanches et samedis pour que les utilisateurs ne puissent choisir que des jours de semaine :

```
my_df.disabledDays = [0, 6];
```

DateField.disabledRanges

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dateFieldInstance.disabledRanges

Description

Propriété qui désactive un jour ou une plage de jours. Cette propriété est un tableau d'objets. Chaque objet du tableau doit être soit un objet Date spécifiant un seul jour à désactiver, soit un objet contenant une des propriétés `rangeStart` ou `rangeEnd`, voire les deux, dont les valeurs doivent être un objet Date. Les propriétés `rangeStart` et `rangeEnd` décrivent les limites de la plage de dates. Si l'une ou l'autre des propriétés est omise, l'étendue est illimitée dans le sens correspondant.

La valeur par défaut de `disabledRanges` est `undefined`.

Indiquez une date complète lorsque vous définissez des dates pour la propriété `disabledRanges`. Par exemple, `new Date(2003,6,24)` plutôt que `new Date()`. Si vous ne spécifiez pas une date complète, l'heure renvoie `00:00:00`.

Exemple

L'exemple suivant définit un tableau comprenant des objets Date `rangeStart` et `rangeEnd` qui désactivent les dates comprises entre le 7 mai et le 7 juin :

```
my_df.disabledRanges = [ {rangeStart: new Date(2003, 4, 7), rangeEnd: new
  Date(2003, 5, 7)}];
```

L'exemple suivant désactive toutes les dates ultérieures au 7 novembre :

```
my_df.disabledRanges = [ {rangeStart: new Date(2003, 10, 7)} ];
```

L'exemple suivant désactive toutes les dates antérieures au 7 octobre :

```
my_df.disabledRanges = [ {rangeEnd: new Date(2002, 9, 7)} ];
```

L'exemple suivant désactive uniquement le 7 décembre :

```
my_df.disabledRanges = [ new Date(2003, 11, 7) ];
```

L'exemple suivant désactive le 7 décembre et le 20 décembre :

```
my_df.disabledRanges = [ new Date(2003, 11, 7), new Date(2003, 11, 20)];
```

DateField.displayedMonth

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`dateFieldInstance``.displayedMonth`

Description

Propriété, nombre indiquant le mois à afficher. Le nombre indique un élément du tableau `monthNames`, 0 correspondant au premier mois. La valeur par défaut est le mois de la date du jour.

Exemple

L'exemple suivant définit le mois affiché sur Décembre :

```
my_df.displayedMonth = 11;
```

Voir aussi

`DateField.displayedYear`

DateField.displayedYear

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`dateFieldInstance``.displayedYear`

Description

Propriété, nombre indiquant l'année affichée. La valeur par défaut est l'année en cours.

Exemple

L'exemple suivant définit l'année affichée sur 2010 :

```
my_df.displayedYear = 2010;
```

Voir aussi

`DateField.displayedMonth`

DateField.firstDayOfWeek

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dateFieldInstance.firstDayOfWeek

Description

Propriété : chiffre indiquant le jour de la semaine (0-6, 0 correspondant au premier élément du tableau `dayNames`) affiché dans la première colonne du composant DateField.
La modification de cette propriété change l'ordre des colonnes des jours, mais n'a aucune incidence sur l'ordre de la propriété `dayNames`. La valeur par défaut est 0 (dimanche).

Exemple

L'exemple suivant définit le premier jour de la semaine sur Lundi :

```
my_df.firstDayOfWeek = 1;
```

Voir aussi

DateField.dayNames

DateField.monthNames

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dateFieldInstance.monthNames

Description

Propriété, tableau de chaînes indiquant les noms des mois en haut du composant DateField. La valeur par défaut est `["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"]`.

Exemple

L'exemple suivant définit les noms de mois pour l'occurrence `my_df` :

```
my_df.monthNames = ["Jan", "Feb","Mar","Apr", "May", "June","July", "Aug",
  "Sept","Oct", "Nov", "Dec"];
```

DateField.open()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`dateFieldInstance``.open()`

Valeur renvoyée

Aucune.

Description

Méthode qui ouvre le sous-composant DateChooser contextuel.

Exemple

Le code suivant ouvre le composant DateChooser contextuel de l'occurrence DateField `my_df` lorsque vous cliquez sur le bouton `my_btn` :

```
// Création d'un objet écouteur.
var btnListener:Object = new Object();
btnListener.click = function() {
   my_df.open();
};

// Ajout de l'écouteur du composant Button.
my_btn.addEventListener("click", btnListener);
```

DateField.open

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.open = function(eventObject:Object) {
  // ...
};
dateFieldInstance.addEventListener("open", listenerObject);
```

Utilisation 2 :

```
on (open) {
  // ...
}
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsqu'un sous-composant DateChooser s'ouvre après que l'utilisateur a cliqué sur l'icône.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence du composant (*dateFieldInstance*) distribue un événement (ici, `open`) qui est géré par une fonction (également appelée *gestionnaire*) dans un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée au déclenchement de l'événement. Lorsqu'il est déclenché, l'événement transmet automatiquement un objet événement (*eventObject*) à la méthode de l'objet écouteur. Chaque objet événement est doté de propriétés contenant des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence DateField. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence du composant. Par exemple, le code suivant, associé au champ de date `my_df`, envoie « _level0.my_df » dans le panneau Sortie :

```
on (open) {
  trace(this);
}
```

Exemple

L'exemple suivant, écrit sur une image du scénario, envoie un message au panneau Sortie lorsqu'un champ de date nommé `my_df` est ouvert. Le message se termine par le numéro d'index du mois en cours. La première ligne de code crée un objet écouteur nommé `dfListener`. La deuxième ligne définit une fonction pour l'événement `open` sur l'objet écouteur. La fonction comporte une instruction `trace()` qui utilise l'objet événement automatiquement transmis à cette fonction (ici `evt_obj`) pour générer un message.
La propriété `target` d'un objet événement est le composant qui a généré l'événement (dans cet exemple, `my_df`). L'utilisateur accède à la propriété `DateField.selectedDate` à partir de la propriété `target` de l'objet événement. La dernière ligne appelle la méthode `EventDispatcher.addEventListener()` à partir de `my_df` et lui transmet l'événement `open` et l'objet écouteur `dfListener` comme paramètres.

```
// Création d'un objet écouteur.
var dfListener:Object = new Object();
dfListener.open = function(evt_obj:Object){
  trace("PullDown Opened" + evt_obj.target.displayedMonth);
}
// Ajout d'un objet écouteur au composant DateField.
my_df.addEventListener("open", dfListener);
```

DateField.pullDown

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`dateFieldInstance``.pullDown`

Description

Propriété (lecture seule), référence au composant DateChooser contenu dans le composant DateField. Le sous-composant DateChooser est instancié lorsqu'un utilisateur clique sur le composant DateField. Néanmoins, si la propriété `pullDown` est référencée avant que l'utilisateur ne clique sur le composant, le sous-composant DateChooser est instancié, puis masqué.

Exemple

L'exemple suivant définit la visibilité du sous-composant DateChooser sur `false`, puis définit sa hauteur et sa largeur sur 300 pixels :

```
my_df.pullDown._visible = false;
my_df.pullDown.setSize(300, 300);
```

DateField.scroll

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.scroll = function(eventObject:Object) {
  // ...
};
dateFieldInstance.addEventListener("scroll", listenerObject);
```

Utilisation 2 :

```
on (scroll) {
  // ...
}
```

Description

Evénement diffusé à tous les écouteurs enregistrés lorsque l'utilisateur clique sur le bouton d'un mois.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence du composant (*dateFieldInstance*) distribue un événement (ici, scroll) géré par une fonction (également appelée *gestionnaire*) associée à un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée au déclenchement de l'événement. Lorsqu'il est déclenché, l'événement transmet automatiquement un objet événement (*eventObject*) à la méthode de l'objet écouteur. Chaque objet événement est doté de propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. L'objet événement de l'événement scroll comprend une propriété supplémentaire, detail, qui peut prendre l'une des valeurs suivantes : nextMonth, previousMonth, nextYear, previousYear.

Pour finir, vous appelez la méthode EventDispatcher.addEventListener() sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel au gestionnaire on() et doit être associé directement à une occurrence DateField. Le mot-clé this, utilisé dans un gestionnaire on() lié à un composant, correspond à l'occurrence du composant. Par exemple, le code suivant, associé au champ de date my_df, envoie « _level0.my_df » dans le panneau Sortie :

```
on (scroll) {
  trace(this);
}
```

Exemple

L'exemple suivant, écrit sur une image du scénario, envoie un message au panneau Sortie lorsque l'utilisateur clique sur le bouton correspondant à un mois sur une occurrence DateField appelée my_df. La première ligne de code crée un objet écouteur nommé dfListener.
La deuxième ligne définit une fonction pour l'événement scroll sur l'objet écouteur.
La fonction comporte une instruction trace() qui utilise l'objet événement automatiquement transmis à cette fonction (ici evt_obj) pour générer un message. La propriété target d'un objet événement est le composant qui a généré l'événement—dans cet exemple, my_df.
La dernière ligne appelle EventDispatcher.addEventListener() à partir de my_df et lui transmet l'événement scroll et l'objet écouteur dfListener comme paramètres.

```
// Création d'un objet écouteur.
var dfListener:Object = new Object();
dfListener.scroll = function(evt_obj:Object) {
 trace(evt_obj.detail);
};

// Ajout d'un objet écouteur au composant DateField.
my_df.addEventListener("scroll", dfListener);
```

DateField.selectableRange

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dateFieldInstance.selectableRange

Description

Propriété qui définit une seule date sélectionnable ou une plage de dates sélectionnables. La valeur de cette propriété est un objet qui comprend deux objets Date appelés `rangeStart` et `rangeEnd`. Les propriétés `rangeStart` et `rangeEnd` désignent les limites de la plage de dates sélectionnables. Si seule la propriété `rangeStart` est définie, toutes les dates ultérieures à `rangeStart` sont activées. Si seule la propriété `rangeEnd` est définie, toutes les dates antérieures à `rangeEnd` sont activées. La valeur par défaut est `undefined`.

Si vous ne souhaitez activer qu'un seul jour, vous pouvez utiliser un objet Date unique comme valeur de la propriété `selectableRange`.

Lorsque vous définissez une date, indiquez la date complète. Par exemple, `new Date(2003,6,24)` plutôt que `new Date()`. Si vous ne spécifiez pas une date complète, l'heure renvoie `00:00:00`.

La valeur de `DateField.selectedDate` est définie sur `undefined` si elle se situe en dehors de la plage sélectionnable.

Les valeurs de `DateField.displayedMonth` et de `DateField.displayedYear` sont définies sur le dernier mois le plus proche dans la plage sélectionnable si le mois en cours n'est pas compris dans cette dernière. Par exemple, si le mois en cours affiché est août et que la plage sélectionnable va de juin à juillet 2003, le mois de juillet 2003 s'affiche.

Exemple

L'exemple suivant définit la plage sélectionnable entre le 7 mai (inclus) et le 7 juin (inclus) :

```
my_df.selectableRange = {rangeStart: new Date(2001, 4, 7), rangeEnd: new
  Date(2003, 5, 7)};
```

L'exemple suivant définit la plage sélectionnable sur les dates ultérieures au 7 mai (inclus) :

```
my_df.selectableRange = {rangeStart: new Date(2003, 4, 7)};
```

L'exemple suivant définit la plage sélectionnable sur les dates antérieures au 7 juin (inclus) :

```
my_df.selectableRange = {rangeEnd: new Date(2003, 5, 7)};
```

L'exemple suivant définit la date sélectionnable uniquement au 7 juin :

```
my_df.selectableRange = new Date(2003, 5, 7);
```

DateField.selectedDate

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dateFieldInstance.selectedDate

Description

Propriété : objet Date qui indique la date sélectionnée si la valeur est comprise dans la plage de valeurs de la propriété `selectableRange`. La valeur par défaut est `undefined`.

Exemple

L'exemple suivant définit la date sélectionnée sur le 7 juin :

```
my_df.selectedDate = new Date(2003, 5, 7);
```

L'exemple suivant utilise une occurrence DateField, appelée `my_df`, sur la scène pour indiquer comment désactiver une date sélectionnée (l'utilisateur peut également cliquer de nouveau pour effacer l'entrée du champ de date) :

```
function dfListener(evt_obj:Object):Void {
  my_df.disabledRanges = [my_df.selectedDate];
}
my_df.addEventListener("change", dfListener);
```

DateField.showToday

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

dateFieldInstance.showToday

Description

Propriété, valeur booléenne qui indique si la date du jour est mise en surbrillance. La valeur par défaut est `true`.

Exemple

L'exemple suivant désactive la mise en surbrillance de la date du jour :

```
my_df.showToday = false;
```

CHAPITRE 16

Classe Delegate

16

Héritage Object > Delegate

Nom de classe ActionScript mx.utils.Delegate

La classe Delegate permet d'exécuter une fonction dans un domaine spécifique. Elle est fournie pour que vous puissiez distribuer le même événement à deux fonctions différentes (reportez-vous à « Délégation d'événements aux fonctions » dans *Utilisation des composants ActionScript 2.0*) et appeler des fonctions au sein du domaine de la classe qui les contient.

REMARQUE	La classe Delegate est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Lorsque vous transmettez une fonction sous forme de paramètre à la méthode `EventDispatcher.addEventListener()`, la fonction est appelée dans le domaine de l'occurrence du composant diffuseur et non de l'objet dans lequel elle est déclarée (voir « Délégation du domaine d'une fonction » dans *Utilisation des composants ActionScript 2.0*). Pour appeler la fonction dans le domaine de l'objet dans lequel elle est déclarée, vous pouvez appeler `Delegate.create()`.

Méthode de la classe Delegate

Le tableau suivant présente la méthode de la classe Delegate.

Méthode	Description
`Delegate.create()`	Méthode statique qui permet d'exécuter une fonction dans un domaine spécifique.

Delegate.create()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
Delegate.create(scopeObject, function)
```

Paramètres

scopeObject Référence à un objet. Il s'agit du domaine dans lequel la fonction doit être exécutée.

function Référence à une fonction.

Description

Méthode (statique) qui permet de déléguer des événements à des domaines et des fonctions spécifiques. Utilisez la syntaxe suivante :

```
import mx.utils.Delegate;
compInstance.addEventListener("eventName", Delegate.create(scopeObject,
  function));
```

Le paramètre *scopeObject* désigne le domaine dans lequel la fonction spécifiée est appelée.

Exemple

Pour consulter des exemples de `Delegate.create()`, reportez-vous à « Délégation des événements » dans *Utilisation des composants ActionScript 2.0*.

Voir aussi

`EventDispatcher.addEventListener()`

CHAPITRE 17

Classe DeltaItem

17

Nom de classe ActionScript mx.data.components.datasetclasses.DeltaItem

La classe DeltaItem fournit des informations sur une opération individuelle effectuée sur un objet de transfert. Elle indique si une propriété d'un objet de transfert a été modifiée directement ou par un appel de méthode. Elle indique également l'état d'origine des propriétés d'un objet de transfert. Par exemple, si la source du paquet delta était un jeu de données, l'objet DeltaItem contient des informations sur tous les champs qui ont été modifiés.

Outre les éléments ci-dessus, un objet DeltaItem contient éventuellement les réponses du serveur, comme la valeur en cours et un message.

REMARQUE	La classe DeltaItem est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Servez-vous de la classe DeltaItem pour accéder aux modifications dans un paquet delta. Pour accéder à ces modifications, utilisez `DeltaPacket.getIterator()`, qui renvoie un itérateur de deltas. Chaque delta contient zéro ou plusieurs occurrences DeltaItem, accessibles via `Delta.getItemByName()` ou `Delta.getChangeList()`.

Propriétés de la classe DeltaItem

Le tableau suivant présente les propriétés de la classe DeltaItem.

Propriété	Description
`DeltaItem.argList`	Lorsqu'une modification est apportée par un appel de méthode, il s'agit du tableau des valeurs transmises à la méthode. Cette propriété est en lecture seule.
`DeltaItem.curValue`	En cas de modification d'une propriété, il s'agit de la valeur de la propriété du serveur en cours. Cette propriété est en lecture seule.
`DeltaItem.delta`	Delta associé à l'objet DeltaItem. Cette propriété est en lecture seule.
`DeltaItem.kind`	Type de modification.
`DeltaItem.message`	Message du serveur associé à l'objet DeltaItem.
`DeltaItem.name`	Nom de la propriété ou de la méthode qui a changé. Cette propriété est en lecture seule.
`DeltaItem.newValue`	En cas de modification d'une propriété, il s'agit de sa nouvelle valeur. Cette propriété est en lecture seule.
`DeltaItem.oldValue`	En cas de modification d'une propriété, il s'agit de son ancienne valeur. Cette propriété est en lecture seule.

DeltaItem.argList

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`deltaitem``.argList`

Description

Propriété (lecture seule), tableau des valeurs transmises à la méthode qui effectue la modification. Cette propriété ne s'applique que si le type de modification est `DeltaItem.Method`.

DeltaItem.curValue

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

deltaitem.curValue

Description

Propriété (lecture seule) ; objet contenant la valeur de la propriété en cours sur la copie du serveur de l'objet de transfert. Cette propriété ne s'applique que si le type de modification est DeltaItem.Property, et elle ne concerne qu'un delta renvoyé par un serveur et est appliquée au jeu de données pour la résolution de l'utilisateur.

DeltaItem.delta

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

deltaitem.delta

Description

Propriété (lecture seule), delta associé à l'objet DeltaItem. Lors de sa création, l'objet DeltaItem est associé à un delta et s'ajoute lui-même à la liste des modifications du delta. Cette propriété fournit une référence au delta auquel cet élément appartient.

DeltaItem.kind

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

deltaitem.kind

Description

Propriété, nombre indiquant le type de modification. Pour évaluer cette propriété, servez-vous des constantes suivantes :

- DeltaItem.Property La modification a été apportée à une propriété sur l'objet de transfert.
- DeltaItem.Method La modification a été apportée via un appel à la méthode sur l'objet de transfert.

DeltaItem.message

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

deltaitem.message

Description

Propriété, chaîne contenant un message du serveur associé à cet objet DeltaItem. Il peut s'agir de n'importe quel message pour la tentative de modification de la propriété ou de l'appel de méthode dans le paquet delta. Ce message concerne généralement un delta renvoyé par un serveur et s'applique au jeu de données pour la résolution.

DeltaItem.name

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

deltaitem.name

Description

Propriété (lecture seule) : chaîne contenant le nom de la propriété modifiée (si le type de modification est DeltaItem.Property) ou le nom de la méthode à l'origine de la modification (si le type de modification est DeltaItem.Method).

DeltaItem.newValue

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

deltaitem.newValue

Description

Propriété (lecture seule), objet contenant la nouvelle valeur de la propriété. Cette propriété ne s'applique que si le type de modification est DeltaItem.Property.

DeltaItem.oldValue

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

deltaitem.oldValue

Description

Propriété (lecture seule), objet contenant l'ancienne valeur de la propriété. Cette propriété ne s'applique que si le type de modification est `DeltaItem.Property`.

CHAPITRE 18

Interface Delta

18

Nom d'interface ActionScript mx.data.components.datasetclasses.Delta

L'interface Delta permet d'accéder à l'objet de transfert, à la collection et aux modifications apportées à l'objet de transfert. Elle vous permet d'accéder aux nouvelles et anciennes valeurs dans un objet de transfert. Par exemple, si le paquet delta provient d'un jeu de données, chaque delta représente alors une ligne ajoutée, modifiée ou supprimée.

REMARQUE	L'interface Delta est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

L'interface Delta fournit également un accès aux messages renvoyés par le processus associé côté serveur. Pour plus d'informations sur les interactions client/serveur, reportez-vous à « Composant RDBMSResolver », à la page 1093.

Servez-vous de l'interface Delta pour examiner le paquet delta avant d'envoyer les modifications au serveur et pour examiner les messages renvoyés par le traitement côté serveur.

Méthodes de l'interface Delta

Le tableau suivant présente les méthodes de l'interface Delta.

Méthode	Description
`Delta.addDeltaItem()`	Ajoute l'occurrence DeltaItem spécifiée.
`Delta.getChangeList()`	Renvoie un tableau des modifications apportées à l'élément en cours.
`Delta.getDeltaPacket()`	Renvoie le paquet delta contenant le delta.
`Delta.getId()`	Renvoie l'identifiant de l'élément en cours au sein de la collection DeltaPacket.
`Delta.getItemByName()`	Renvoie l'objet DeltaItem spécifié.

Méthode	Description
`Delta.getMessage()`	Renvoie le message associé à l'élément en cours.
`Delta.getOperation()`	Renvoie l'opération exécutée sur l'élément en cours dans la collection d'origine.
`Delta.getSource()`	Renvoie l'objet de transfert auquel les modifications ont été apportées.

Delta.addDeltaItem()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`delta``.addDeltaItem(`*`deltaitem`*`)`

Paramètres

`deltaitem` Occurrence DeltaItem à ajouter à ce delta.

Renvoie

Aucune.

Description

Méthode qui ajoute l'occurrence DeltaItem spécifiée. Si l'occurrence existe déjà, cette méthode la remplace.

Exemple

L'exemple suivant appelle la méthode `addDeltaItem()` :

```
//...
var d:Delta = new DeltaImpl("ID1345678", curItem, DeltaPacketConsts.Added,
  "", false);
d.addDeltaItem(new DeltaItem(DeltaItem.Property, "ID", {oldValue:15,
  newValue:16}));
//...
```

Delta.getChangeList()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

delta.getChangeList()

Paramètres

Aucun.

Renvoie

Un tableau des occurrences DeltaItem associées.

Description

Méthode qui renvoie un tableau des occurrences DeltaItem associées. Chaque occurrence DeltaItem du tableau décrit une modification apportée à l'élément.

Exemple

L'exemple suivant appelle la méthode `getChangeList()` :

```
//...
case mx.data.components.datasetclasses.DeltaPacketConsts.Modified: {
  // dpDelta est une variable de type Delta.
  var changes:Array = dpDelta.getChangeList();
  for(var i:Number = 0; i<changes.length; i++) {
    // getChangeMessage est une méthode définie par l'utilisateur.
    changeMsg = _parent.getChangeMessage(changes[i]);
    trace(changeMsg);
  }
//...
```

Delta.getDeltaPacket()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

delta.getDeltaPacket()

Paramètres

Aucun.

Renvoie

Le paquet delta qui contient ce delta.

Description

Méthode qui renvoie le paquet delta contenant ce delta. Cette méthode vous permet d'écrire du code capable de traiter les paquets delta de façon générique au niveau du delta.

Exemple

L'exemple suivant utilise la méthode `getDeltaPacket()` pour accéder à la source de données du paquet delta :

```
while(dpCursor.hasNext()) {
  dpDelta = Delta(dpCursor.next());
  trace("DeltaPacket source is: " + dpDelta.getDeltaPacket().getSource());
}
```

Delta.getId()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

delta.getId()

Paramètres

Aucun.

Renvoie

Un objet, l'identifiant unique de cet élément dans la collection DeltaPacket.

Description

Méthode qui renvoie l'identifiant unique de cet élément dans la collection DeltaPacket. Utilisez cet identifiant dans le composant source du paquet delta pour recevoir les mises à jour et modifier les éléments à partir duquel le paquet delta a été généré. Supposons par exemple que le composant DataSet envoie des mises à jour à un serveur qui renvoie à son tour de nouvelles valeurs de champs clés. Cette méthode permet alors au composant DataSet d'examiner le paquet delta résultant, de trouver l'objet de transfert d'origine et de lui appliquer les mises à jour nécessaires.

Exemple

L'exemple suivant appelle la méthode `getId()` :

```
while(dpCursor.hasNext()) {
  dpDelta = Delta(dpCursor.next());
  trace("id ["+dpDelta.getId()+"]");
}
```

Delta.getItemByName()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`delta``.getItemByName(`*`name`*`)`

Paramètres

`name` Chaîne qui spécifie le nom de la propriété ou de la méthode pour l'objet DeltaItem associé.

Renvoie

Objet DeltaItem spécifié par *`name`*. Si aucun objet DeltaItem correspondant à *`name`* n'est trouvé, la méthode renvoie `null`.

Description

Méthode : renvoie l'objet DeltaItem spécifié par *name*. Lorsque des appels de méthode ou des changements de propriétés sont nécessaires par nom sur un objet de transfert, cette méthode constitue l'accès le plus efficace.

Exemple

L'exemple suivant appelle la méthode `getItemByName()` :

```
private function buildFieldTag(deltaObj:Delta, field:Object,
  isKey:Boolean):String {
  var chgItem:DeltaItem = deltaObj.getItemByName(field.name);
  var result:String= "<field name=\"" + field.name + "\" type=\"" +
  field.type.name + "\"";
  var oldValue:String;
  var newValue:String;
  if (deltaObj.getOperation() != DeltaPacketConsts.Added) {
    oldValue = (chgItem != null ? (chgItem.oldValue != null ?
  encodeFieldValue(field.name, chgItem.oldValue) : __nullValue) :
  encodeFieldValue(field.name, deltaObj.getSource()[field.name]));
    newValue = (chgItem.newValue != null ? encodeFieldValue(field.name,
  chgItem.newValue) : __nullValue);
    result+= " oldValue=\""  + oldValue + "\"";
    result+= chgItem != null ? " newValue=\"" + newValue + "\"" : "";
    result+= " key=\"" + isKey.toString() + "\" />";
  }
  else {
    result+= " newValue=\"" +encodeFieldValue(field.name,
  deltaObj.getSource()[field.name]) + "\"";
    result+= " key=\"" + isKey.toString() + "\" />";
  }
  return result;
}
```

Delta.getMessage()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
delta.getMessage()
```

Paramètres

Aucun.

Renvoie

Chaîne : message associé au *delta*.

Description

Méthode qui renvoie le message associé à ce delta. En général, ce message n'est renseigné que si le paquet delta a été renvoyé par un serveur en réponse à des tentatives de mises à jour. Pour plus d'informations, voir « Composant RDBMSResolver », à la page 1093.

Exemple

L'exemple suivant appelle la méthode `getMessage()` :

```
//...
var dpi:Iterator = dp.getIterator();
var d:Delta;
while(dpi.hasNext()) {
  d= dpi.next();
  trace(d.getMessage());
}
//...
```

Delta.getOperation()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

delta`.getOperation()`

Paramètres

Aucun.

Renvoie

Un nombre, l'opération exécutée sur l'élément dans la collection d'origine.

Description

Méthode qui renvoie l'opération exécutée sur cet élément dans la collection d'origine. Les valeurs valides sont `DeltaPacketConsts.Added`, `DeltaPacketConsts.Removed` et `DeltaPacketConsts.Modified`.

Vous devez importer mx.data.components.datasetclasses.DeltaPacketConsts ou définir entièrement chaque constante.

Exemple

L'exemple suivant appelle la méthode `getOperation()` :

```
while(dpCursor.hasNext()) {
  dpDelta = Delta(dpCursor.next());
  op=dpDelta.getOperation();
  trace("DeltaPacket source is: " + dpDelta.getDeltaPacket().getSource());
  switch(op) {
    case mx.data.components.datasetclasses.DeltaPacketConsts.Added:
      trace("***In case DeltaPacketConsts.Added ***");
    case mx.data.components.datasetclasses.DeltaPacketConsts.Modified: {
      trace("***In case DeltaPacketConsts.Modified ***");
  }
}
```

Delta.getSource()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`delta``.getSource()`

Paramètres

Aucun.

Renvoie

L'objet de transfert qui a subi les modifications.

Description

Méthode qui renvoie l'objet de transfert qui a subi les modifications.

Exemple

L'exemple suivant appelle la méthode `getSource()` :

```
while(dpCursor.hasNext()) {
  dpDelta = Delta(dpCursor.next());
  op=dpDelta.getOperation();
  switch(op) {
    case mx.data.components.datasetclasses.DeltaPacketConsts.Modified: {
    // Les valeurs d'origine sont
    trace("Unmodified source is: ");
    var src = dpDelta.getDeltaPacket().getSource();
    for(var i in src){
      if(typeof(src[i]) != "function"){
        trace(i+"="+src[i]);
      }
    }
  }
}
```

CHAPITRE 19

Interface DeltaPacket

19

Nom d'interface ActionScript mx.data.components.datasetclasses.DeltaPacket

L'interface DeltaPacket est fournie par la propriété `deltaPacket` du composant DataSet, qui fait lui-même partie des fonctionnalités de gestion des données. En général, le paquet delta est utilisé en interne par les composants Resolver. L'interface DeltaPacket, l'interface Delta associée et la classe DeltaItem permettent de gérer les modifications apportées aux données. Ces composants ne sont pas visibles à l'exécution.

REMARQUE	L'interface DeltaPacket est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Un paquet delta est un ensemble d'instructions optimisé décrivant toutes les modifications apportées aux données d'un jeu de données. Lorsque la méthode `DataSet.applyUpdates()` est appelée, le composant DataSet renseigne la propriété `DataSet.deltaPacket`. En général, cette propriété est reliée (par liaison de données) à un composant Resolver, tel que RDBMSResolver. Ce composant Resolver convertit le paquet delta en paquet de mise à jour au format approprié.

CONSEIL	A moins que vous n'écriviez votre propre composant Resolver personnalisé, vous n'aurez probablement jamais besoin de connaître ou d'écrire le code qui permet d'accéder aux méthodes ou aux propriétés d'un paquet delta.

Un paquet delta contient un ou plusieurs deltas (voir « Interface Delta », à la page 485), et chaque delta contient zéro ou plusieurs éléments delta (voir « Classe DeltaItem », à la page 479).

Méthodes de l'interface DeltaPacket

Le tableau suivant présente les méthodes de l'interface DeltaPacket.

Méthode	Description
`DeltaPacket.getConfigInfo()`	Renvoie les informations de configuration spécifiques à l'implémentation de l'interface DeltaPacket.
`DeltaPacket.getIterator()`	Renvoie l'itérateur du paquet delta répété à travers la liste des deltas du paquet delta.
`DeltaPacket.getSource()`	Renvoie la source du paquet delta. Il s'agit du composant qui a exposé ce paquet delta.
`DeltaPacket.getTimestamp()`	Renvoie la date et l'heure à laquelle le paquet delta a été instancié.
`DeltaPacket.getTransactionId()`	Renvoie l'identifiant de transaction de ce paquet delta.
`DeltaPacket.logChanges()`	Indique si l'utilisateur du paquet delta doit enregistrer les modifications qu'il spécifie.

DeltaPacket.getConfigInfo()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`deltaPacket``.getConfigInfo(`*`info`*`)`

Propriétés

`info` Objet qui contient des informations spécifiques à l'implémentation.

Renvoie

Un objet contenant les informations requises pour l'implémentation DeltaPacket spécifique.

Description

Méthode qui renvoie les informations de configuration spécifiques à l'implémentation de l'interface DeltaPacket. Cette méthode permet aux implémentations de l'interface DeltaPacket d'accéder aux informations personnalisées.

Exemple

L'exemple suivant appelle la méthode `getConfigInfo()` :

```
// ...
new DeltaPacketImpl(source, getTransactionId(), null, logChanges(),
  getConfigInfo());
// ...
```

DeltaPacket.getIterator()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`deltaPacket``.getIterator()`

Paramètres

Aucun.

Renvoie

Une interface à l'itérateur de la collection DeltaPacket répétée à travers la liste des deltas du paquet delta.

Description

Méthode qui renvoie l'itérateur de la collection DeltaPacket. Cette méthode est un moyen d'effectuer une boucle à travers les modifications du paquet delta. Pour consulter la description complète de cette interface, reportez-vous à « Interface Iterator », à la page 777.

Exemple

L'exemple suivant utilise la méthode `getIterator()` pour accéder à l'itérateur des deltas d'un paquet delta et une instruction `while` pour effectuer une boucle à travers les deltas :

```
var deltapkt:DeltaPacket = _parent.myDataSet.deltaPacket;
trace("*** Test deltapacket. Trans ID is: " + deltapkt.getTransactionId() +
  " ***");
var OPS:Array = new Array("added", "removed", "modified");
var dpCursor:Iterator = deltapkt.getIterator();
var dpDelta:Delta;
var op:Number;
var changeMsg:String;
while(dpCursor.hasNext()) {
  dpDelta = Delta(dpCursor.next());
  op=dpDelta.getOperation();
}
```

DeltaPacket.getSource()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`deltaPacket``.getSource()`

Paramètres

Aucun.

Renvoie

Un objet, source de la collection DeltaPacket. Cet objet est généralement un descendant de MovieClip, mais ce n'est pas obligatoire. Par exemple, si la source est un jeu de données, cet objet peut être `_level0.myDataSet`.

Description

Méthode qui renvoie la source de la collection DeltaPacket.

Exemple

L'exemple suivant appelle la méthode `getSource()` :

```
// ...
var deltapkt:DeltaPacket = _parent.myDataSet.deltaPacket;
var dpSourceText:String = "Source: " + deltapkt.getSource();
trace(dpSourceText);
// ...
```

DeltaPacket.getTimestamp()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

deltaPacket.getTimestamp()

Paramètres

Aucun.

Renvoie

Un objet Date contenant la date et l'heure de création du paquet delta.

Description

Méthode qui renvoie la date et l'heure de création du paquet delta.

Exemple

L'exemple suivant appelle la méthode `getTimestamp()` :

```
// ...
var deltapkt:DeltaPacket = _parent.myDataSet.deltaPacket;
var dpTime:String = " Time: " + deltapkt.getTimestamp();
trace(dpTime);
// ...
```

DeltaPacket.getTransactionId()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

deltaPacket.getTransactionId()

Paramètres

Aucun.

Renvoie

Une chaîne, identifiant de transaction unique pour le regroupement des transactions de paquets delta.

Description

Méthode qui renvoie l'identifiant de transaction du paquet delta. Cet identifiant unique permet de regrouper une transaction envoi/réception pour un paquet delta. Le jeu de données l'utilise pour déterminer si le paquet delta fait partie de la même transaction dont il est originaire avec l'appel `DataSet.applyUpdates()`.

Exemple

L'exemple suivant appelle la méthode `getTransactionId()` :

```
// ...
var deltapkt:DeltaPacket = _parent.myDataSet.deltaPacket;
trace("*** Trans ID is: " + deltapkt.getTransactionId() + " ***");
// ...
```

DeltaPacket.logChanges()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`deltaPacket``.logChanges()`

Paramètres

Aucun.

Renvoie

Valeur booléenne `true` si l'utilisateur du paquet delta doit enregistrer les modifications détectées dans le paquet delta.

Description

Méthode : renvoie `true` si l'utilisateur de ce paquet delta doit enregistrer les modifications qu'il spécifie. Cette valeur est avant tout utilisée pour communiquer les modifications intervenant entre les jeux de données via des objets partagés ou d'un serveur à un jeu de données local. Dans les deux cas, le jeu de données ne doit pas enregistrer les modifications spécifiées.

Exemple

L'exemple suivant appelle la méthode `logChanges()` :

```
var deltapkt:DeltaPacket = _parent.myDataSet.deltaPacket;
if(deltapkt.logChanges()) {
  trace("*** We need to log changes. ***");
}
else {
  trace("*** We do not need to log changes");
}
```

CHAPITRE 20

Classe DepthManager

20

Nom de classe ActionScript mx.managers.DepthManager

La classe DepthManager permet de gérer les affectations de profondeur relatives des composants ou des clips, dont `_root`. Elle autorise également à gérer les profondeurs réservées dans un clip spécial et ceci à la profondeur la plus élevée sur `_root` pour les services de niveau système, tels que le curseur et les info-bulles.

REMARQUE La classe DepthManager est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

En général, Depth Manager gère les composants automatiquement, à l'aide de son propre algorithme de « réarrangement ». Si vous n'êtes pas expérimenté en développement Flash, il n'est pas nécessaire d'utiliser ses API.

CONSEIL Pour utiliser la classe DepthManager pour des occurrences de clip, la bibliothèque ou la scène doit contenir un composant et vous devez utiliser « import mx.managers.DepthManager » au début de votre ActionScript.

L'API de tri de profondeur relative se compose des méthodes suivantes :

- `DepthManager.createChildAtDepth()`
- `DepthManager.createClassChildAtDepth()`
- `DepthManager.setDepthAbove()`
- `DepthManager.setDepthBelow()`
- `DepthManager.setDepthTo()`

Les méthodes suivantes composent les API d'espace de profondeur réservée :

- `DepthManager.createClassObjectAtDepth()`
- `DepthManager.createObjectAtDepth()`

Méthodes de la classe DepthManager

Le tableau suivant présente les méthodes de la classe DepthManager.

Méthode	Description
`DepthManager.createChildAtDepth()`	Crée un enfant du symbole indiqué à la profondeur spécifiée.
`DepthManager.createClassChildAtDepth()`	Crée un objet de la classe indiquée à la profondeur spécifiée.
`DepthManager.createClassObjectAtDepth()`	Crée une occurrence de la classe indiquée à la profondeur spécifiée dans le clip spécial de profondeur la plus élevée.
`DepthManager.createObjectAtDepth()`	Crée un objet à une profondeur spécifiée dans le clip spécial de profondeur la plus élevée.
`DepthManager.setDepthAbove()`	Définit la profondeur au-dessus de l'occurrence spécifiée.
`DepthManager.setDepthBelow()`	Définit la profondeur sous l'occurrence spécifiée.
`DepthManager.setDepthTo()`	Définit la profondeur de l'occurrence spécifiée dans le clip de profondeur la plus élevée.

Propriétés de la classe DepthManager

Le tableau suivant présente les propriétés de la classe DepthManager. Les valeurs constantes indiquées sont les valeurs par défaut utilisées par l'algorithme DepthManager pour organiser la profondeur. Si vous suivez les propriétés suivantes, ces valeurs constantes s'affichent dans le panneau Sortie.

Néanmoins, une fois que vous avez implémenté une méthode DepthManager (`DepthManager.setDepthTo()`, par exemple) à l'aide des propriétés suivantes et que vous avez tracé la profondeur du clip ou du composant, DepthManager définit les profondeurs par incréments de 20. L'algorithme incrémente les profondeurs au cas où Flash aurait besoin d'insérer un autre élément au milieu, selon d'autres scripts, composants, etc.

Propriété	Description
`DepthManager.kBottom`	Propriété statique de valeur constante 202.
`DepthManager.kCursor`	Propriété statique de valeur constante 101. Il s'agit de la profondeur du curseur.
`DepthManager.kNotopmost`	Propriété statique de valeur constante 204.

Propriété	Description
`DepthManager.kTooltip`	Propriété statique de valeur constante 102. Il s'agit de la profondeur de l'info-bulle.
`DepthManager.kTop`	Propriété statique de valeur constante 201.
`DepthManager.kTopmost`	Propriété statique de valeur constante 203.

DepthManager.createChildAtDepth()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

movieClipInstance.createChildAtDepth(*linkageName*, *depthFlag*[, *initObj*])

Paramètres

linkageName Identifiant de liaison. Ce paramètre est une chaîne.

depthFlag Une des valeurs suivantes : `DepthManager.kTop`, `DepthManager.kBottom`, `DepthManager.kTopmost`, `DepthManager.kNotopmost`. Tous les indicateurs de profondeur sont des propriétés statiques de la classe DepthManager. Vous devez référencer le package DepthManager (`mx.managers.DepthManager.kTopmost`, par exemple) ou utiliser l'instruction `import` pour importer le package DepthManager.

initObj Objet d'initialisation. Ce paramètre est facultatif.

Valeur renvoyée

Une référence à l'objet créé. Le type renvoyé est MovieClip.

Description

Méthode : crée une occurrence enfant du symbole spécifié par le paramètre *linkageName* à la profondeur spécifiée par le paramètre *depthFlag*.

Exemple

L'exemple suivant crée une occurrence `minuteHand` du clip MinuteSymbol et la place devant l'horloge :

```
import mx.managers.DepthManager;
minuteHand = clock.createChildAtDepth("MinuteSymbol", DepthManager.kTop);
```

DepthManager.createClassChildAtDepth()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

movieClipInstance.createClassChildAtDepth(*className*, *depthFlag*[, *initObj*])

Paramètres

className Nom de classe. Ce paramètre est de type Function.

depthFlag Une des valeurs suivantes : DepthManager.kTop, DepthManager.kBottom, DepthManager.kTopmost, DepthManager.kNotopmost. Tous les indicateurs de profondeur sont des propriétés statiques de la classe DepthManager. Vous devez référencer le package DepthManager (mx.managers.DepthManager.kTopmost, par exemple) ou utiliser l'instruction import pour importer le package DepthManager.

initObj Objet d'initialisation. Ce paramètre est facultatif.

Valeur renvoyée

Une référence à l'enfant créé. Le type renvoyé est UIObject.

Description

Méthode : crée un enfant de la classe spécifiée par le paramètre *className* à la profondeur spécifiée par le paramètre *depthFlag*.

Exemple

Le code suivant trace un rectangle de focus face à tous les objets NoTopmost :

```
import mx.managers.DepthManager
this.ring = createClassChildAtDepth(mx.skins.RectBorder,
  DepthManager.kTop);
```

Le code suivant crée une occurrence de la classe Button et lui transmet une valeur pour sa propriété label en tant que paramètre *initObj* :

```
import mx.managers.DepthManager
button1 = createClassChildAtDepth(mx.controls.Button, DepthManager.kTop,
  {label: "Top Button"});
```

DepthManager.createClassObjectAtDepth()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
DepthManager.createClassObjectAtDepth(className, depthSpace[, initObj])
```

Paramètres

`className` Nom de classe. Ce paramètre est de type Function.

`depthSpace` Une des valeurs suivantes : `DepthManager.kCursor`, `DepthManager.kTooltip`. Tous les indicateurs de profondeur sont des propriétés statiques de la classe DepthManager. Vous devez référencer le package DepthManager (`mx.managers.DepthManager.kCursor`, par exemple) ou utiliser l'instruction `import` pour importer le package DepthManager.

`initObj` Objet d'initialisation. Ce paramètre est facultatif.

Valeur renvoyée

Une référence à l'objet créé. Le type renvoyé est UIObject.

Description

Méthode : crée un objet de la classe spécifiée par le paramètre *`className`* à la profondeur spécifiée par le paramètre *`depthSpace`*. Cette méthode est utilisée pour accéder aux espaces de profondeur réservés dans le clip spécial de profondeur la plus élevée.

Exemple

L'exemple suivant crée un objet à partir de la classe Button :

```
import mx.managers.DepthManager
myCursorButton = DepthManager.createClassObjectAtDepth(mx.controls.Button,
  DepthManager.kCursor, {label: "Cursor"});
```

DepthManager.createObjectAtDepth()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
DepthManager.createObjectAtDepth(linkageName, depthSpace[, initObj])
```

Paramètres

linkageName Identifiant de liaison. Ce paramètre est de type String.

depthSpace Une des valeurs suivantes : `DepthManager.kCursor`, `DepthManager.kTooltip`. Tous les indicateurs de profondeur sont des propriétés statiques de la classe DepthManager. Vous devez référencer le package DepthManager (`mx.managers.DepthManager.kCursor`, par exemple) ou utiliser l'instruction `import` pour importer le package DepthManager.

initObj Objet d'initialisation facultatif.

Valeur renvoyée

Une référence à l'objet créé. Le type renvoyé est MovieClip.

Description

Méthode qui crée un objet à la profondeur spécifiée. Cette méthode est utilisée pour accéder aux espaces de profondeur réservés dans le clip spécial de profondeur la plus élevée.

Exemple

L'exemple suivant crée une occurrence du symbole TooltipSymbol et la place à la profondeur réservée aux info-bulles :

```
import mx.managers.DepthManager
myCursorTooltip = DepthManager.createObjectAtDepth("TooltipSymbol",
  DepthManager.kTooltip);
```

DepthManager.kBottom

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
DepthManager.kBottom
```

Description

Propriété (statique) dont la valeur constante est 202. Cette propriété est transmise en tant que paramètre dans les appels à `DepthManager.createClassChildAtDepth()` et `DepthManager.createChildAtDepth()` pour placer un contenu derrière un autre.

DepthManager.kCursor

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
DepthManager.kCursor
```

Description

Propriété (statique) dont la valeur constante est 101. Cette propriété est transmise en tant que paramètre dans les appels à `DepthManager.createClassObjectAtDepth()` et `DepthManager.createObjectAtDepth()` pour demander un placement à la profondeur du curseur.

DepthManager.kNotopmost

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
DepthManager.kNotopmost
```

Description

Propriété (statique) dont la valeur constante est 204. Cette propriété est transmise en tant que paramètre dans les appels à `DepthManager.createClassChildAtDepth()` et `DepthManager.createChildAtDepth()` pour demander un retrait du niveau supérieur.

DepthManager.kTooltip

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
DepthManager.kTooltip
```

Description

Propriété (statique) dont la valeur constante est 102. Cette propriété est transmise en tant que paramètre dans les appels à `DepthManager.createClassObjectAtDepth()` et `DepthManager.createObjectAtDepth()` pour placer un objet à la profondeur de l'info-bulle.

DepthManager.kTop

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
DepthManager.kTop
```

Description

Propriété (statique) dont la valeur constante est 201. Cette propriété est transmise en tant que paramètre dans les appels à `DepthManager.createClassChildAtDepth()` et `DepthManager.createChildAtDepth()` pour demander le placement sur un autre contenu, mais sous `DepthManager.kTopmost`.

DepthManager.kTopmost

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
DepthManager.kTopmost
```

Description

Propriété (statique) dont la valeur constante est 203. Cette propriété est utilisée dans les appels à `DepthManager.createClassChildAtDepth()` et `DepthManager.createChildAtDepth()` pour demander le placement sur un autre contenu, y compris des objets `DepthManager.kTop`.

DepthManager.setDepthAbove()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
movieClipInstance.setDepthAbove(instance)
```

Paramètres

instance Nom d'occurrence. Ce paramètre est de type MovieClip.

Valeur renvoyée

Aucune.

Description

Méthode : définit la profondeur d'une occurrence de clip ou de composant au-dessus de la profondeur de l'occurrence spécifiée par le paramètre *instance* en déplaçant au besoin d'autres objets.

DepthManager.setDepthBelow()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004 ou Flash MX Professional 2004.

Utilisation

```
movieClipInstance.setDepthBelow(instance)
```

Paramètres

instance Nom d'occurrence. Ce paramètre est de type MovieClip.

Valeur renvoyée

Aucune.

Description

Méthode qui définit la profondeur d'une occurrence de clip ou de composant sous la profondeur de l'occurrence spécifiée, en déplaçant au besoin d'autres objets.

Exemple

Le code suivant définit la profondeur de l'occurrence `textInput` sous la profondeur de `button` :

```
textInput.setDepthBelow(button);
```

DepthManager.setDepthTo()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`movieClipInstance``.setDepthTo(`*`depthFlag`*`)`

Paramètres

`depthFlag` Une des valeurs suivantes : `DepthManager.kTop`, `DepthManager.kBottom`, `DepthManager.kTopmost`, `DepthManager.kNotopmost`. Tous les indicateurs de profondeur sont des propriétés statiques de la classe DepthManager. Vous devez référencer le package DepthManager (`mx.managers.DepthManager.kTopmost`, par exemple) ou utiliser l'instruction `import` pour importer le package DepthManager.

Valeur renvoyée

Aucune.

Description

Méthode : définit la profondeur de *`movieClipInstance`* sur la valeur spécifiée par *`depthFlag`*. Cette méthode place une occurrence à une autre profondeur afin de faire de la place pour un autre objet. DepthManager utilise un algorithme de « réarrangement » pour définir les profondeurs par incréments de 20. L'algorithme incrémente les profondeurs au cas où Flash aurait besoin d'insérer un autre élément au milieu, selon d'autres scripts, composants, etc.

Exemple

L'exemple suivant utilise deux composants (ou clips) pour augmenter leur profondeur en alternance, par incréments de 20, chaque fois que vous cliquez sur chacun d'eux. Ajoutez d'abord un composant Button sur la scène et nommez l'occurrence `a_btn`. Ajoutez ensuite un autre composant Button sur la scène et nommez l'occurrence `b_btn`. Vérifiez que les boutons se chevauchent de la façon suivante :


```
import mx.managers.DepthManager;

a_btn.onRelease = function() {
b_btn.setDepthTo(DepthManager.kTop);
var b_depth:Number = b_btn.getDepth();
trace(b_depth);
}

b_btn.onRelease = function() {
a_btn.setDepthTo(DepthManager.kTop);
var a_depth:Number = a_btn.getDepth();
trace(a_depth);
}
```

Testez le fichier SWF. Lorsque vous cliquez sur le bouton supérieur, l'autre bouton change de profondeur et se déplace à l'avant, et le panneau Sortie affiche la profondeur du bouton en question. Les valeurs sont les suivantes : 20, puis 40, puis 60, par incrément de 20 chaque fois que vous cliquez.

REMARQUE	Si vous utilisez DepthManager avec des occurrences de clip au lieu d'occurrences de composant, il se peut que vous deviez ajoutez un composant d'interface utilisateur à votre bibliothèque (si ce n'est pas déjà fait) pour que DepthManager fonctionne correctement. Pour que DepthManager fonctionne correctement, un composant doit se trouver sur la scène ou dans la bibliothèque.

Pour plus d'informations sur l'ordre des piles et des profondeurs, reportez-vous à la section *Définition de la profondeur maximale disponible suivante* dans le guide *Formation à ActionScript 2.0 dans Adobe Flash*.

CHAPITRE 21

Classe EventDispatcher

21

Les événements signalent à votre application quand l'utilisateur interagit avec un composant et quand d'importantes modifications affectent l'aspect ou le cycle de vie d'un composant, par exemple sa création, sa destruction ou son redimensionnement.

REMARQUE

La classe EventDispatcher est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Les méthodes de la classe EventDispatcher vous permettent d'ajouter et de supprimer des écouteurs d'événements de sorte que votre code puisse réagir de façon appropriée aux événements. Par exemple, la méthode `EventDispatcher.addEventListener()` permet d'enregistrer un écouteur avec une occurrence de composant. L'écouteur est invoqué lorsque l'événement d'un composant est déclenché.

Lorsque vous désirez écrire un objet personnalisé émettant des événements non liés à l'interface utilisateur, EventDispatcher se révèle plus léger et plus rapide à utiliser sous forme de mix-in de UIComponent que UIEventDispatcher.

Objets événement

Un objet événement est transmis à un écouteur sous forme de paramètre. Il s'agit d'un objet ActionScript dont les propriétés contiennent des informations sur l'événement qui s'est produit. Vous pouvez l'utiliser à l'intérieur de la fonction de rappel de l'écouteur pour déterminer le nom de l'événement diffusé ou le nom d'occurrence du composant qui a diffusé l'événement. Le code suivant, par exemple, utilise la propriété `target` de l'objet événement `evtObj` pour accéder à la propriété `label` de l'occurrence `myButton` et envoyer la valeur vers le panneau Sortie :

```
listener = new Object();
listener.click = function(evtObj){
  trace("The " + evtObj.target.label + " button was clicked");
}
myButton.addEventListener("click", listener);
```

Certaines propriétés d'objet événement sont définies dans les spécifications du W3C (www.w3.org/TR/DOM-Level-3-Events/events.html), mais ne sont pas implémentées dans la version 2 de l'architecture des composants Adobe. Tous les objets événements possèdent les propriétés énumérées dans le tableau suivant. Des propriétés supplémentaires sont définies pour certains événements. Dans ce cas, elles sont indiquées dans l'entrée correspondant à l'événement.

Propriété	Description
`type`	Chaîne contenant le nom de l'événement.
`target`	Référence à l'occurrence de composant qui émet l'événement.

Classe EventDispatcher (API)

Nom de classe ActionScript mx.events.EventDispatcher

Méthodes de la classe EventDispatcher

Le tableau suivant présente les méthodes de la classe EventDispatcher.

Méthode	Description
`EventDispatcher.addEventListener()`	Associe un écouteur à une occurrence de composant.
`EventDispatcher.dispatchEvent()`	Distribue un événement par programmation.
`EventDispatcher.removeEventListener()`	Supprime un écouteur d'événement d'une occurrence de composant.

EventDispatcher.addEventListener()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004 ou Flash MX Professional 2004.

Utilisation

componentInstance.addEventListener(*event*, *listener*)

Paramètres

event Chaîne correspondant au nom de l'événement.

listener Référence à une fonction ou à un objet écouteur.

Valeur renvoyée

Aucune.

Description

Méthode qui associe un objet écouteur à une occurrence de composant qui diffuse un événement. Lorsque l'événement est déclenché, l'objet écouteur ou la fonction est averti(e). Vous pouvez appeler cette méthode à partir de toute occurrence de composant. Le code suivant, par exemple, enregistre un écouteur dans l'occurrence du composant `myButton` :

```
myButton.addEventListener("click", myListener);
```

Vous devez définir l'écouteur en tant qu'objet ou fonction avant d'appeler `addEventListener()` pour enregistrer l'écouteur avec l'occurrence de composant. Si l'écouteur est un objet, il doit avoir une fonction de rappel, invoquée lorsque l'événement se produit. En général, cette fonction de rappel porte le même nom que l'événement associé à l'écouteur. Si l'écouteur est une fonction, celle-ci est invoquée lorsque l'événement se produit. Pour plus d'informations, consultez « Gestion des événements à l'aide d'écouteurs » dans *Utilisation des composants ActionScript 2.0*.

Vous pouvez enregistrer plusieurs écouteurs dans une même occurrence de composant, mais vous devez appeler `addEventListener()` séparément pour chacun d'eux. Vous pouvez également enregistrer un écouteur dans plusieurs occurrences de composant, mais vous devez appeler `addEventListener()` séparément pour chacune d'elles. Le code suivant, par exemple, définit un objet écouteur et l'affecte à deux occurrences du composant Button, dont les propriétés `label` sont respectivement `button1` et `button2` :

```
lo = new Object();
lo.click = function(evt){
  trace(evt.target.label + " clicked");
}
button1.addEventListener("click", lo);
button2.addEventListener("click", lo);
```

L'ordre d'exécution n'est pas garanti. Vous ne pouvez pas anticiper l'ordre dans lequel les écouteurs seront appelés.

Un objet événement est transmis à l'écouteur sous forme de paramètre. L'objet événement a des propriétés contenant des informations sur l'événement qui s'est produit. Vous pouvez utiliser l'objet événement à l'intérieur de la fonction de rappel de l'écouteur pour connaître le type d'événement qui s'est produit et l'occurrence qui a diffusé l'événement. Dans l'exemple ci-dessus, l'objet événement correspond à `evt` (vous pouvez utiliser n'importe quel identifiant comme nom d'objet événement) et il est utilisé dans les instructions `if` pour connaître l'occurrence Button sur laquelle l'utilisateur a cliqué. Pour plus d'informations, consultez « Objet événement » dans *Utilisation des composants ActionScript 2.0*.

Exemple

L'exemple suivant définit un objet écouteur, `myListener`, et définit la fonction de rappel de l'événement `click`. Il appelle ensuite `addEventListener()` pour enregistrer l'objet écouteur `myListener` avec l'occurrence du composant `myButton`.

```
myListener = new Object();
myListener.click = function(evt){
  trace(evt.type + " triggered");
}
myButton.addEventListener("click", myListener);
```

Pour tester ce code, placez sur la scène une occurrence du composant Button nommée `myButton`, puis insérez ce code dans l'image 1.

EventDispatcher.dispatchEvent()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004 ou Flash MX Professional 2004.

Utilisation

```
dispatchEvent(eventObject)
```

Paramètres

`eventObject` Référence à un objet événement. L'objet événement doit disposer d'une propriété `type` correspondant à une chaîne indiquant le nom de l'événement. En général, l'objet événement présente également une propriété `target` correspondant au nom de l'occurrence qui diffuse l'événement. Vous pouvez définir d'autres propriétés dans l'objet événement pour aider l'utilisateur à mieux comprendre l'événement lors de sa diffusion.

Valeur renvoyée

Aucune.

Description

Méthode qui distribue un événement à n'importe quel écouteur associé à une occurrence de la classe. Cette méthode est généralement appelée depuis un fichier de classe de composant. Par exemple, le fichier de classe SimpleButton.as distribue l'événement `click`.

Exemple

L'exemple suivant distribue un événement `click` :

```
dispatchEvent({type:"click"});
```

EventDispatcher.removeEventListener()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004 ou Flash MX Professional 2004.

Utilisation

```
componentInstance.removeEventListener(event, listener)
```

Paramètres

event Chaîne correspondant au nom de l'événement.

listener Référence à une fonction ou à un objet écouteur.

Valeur renvoyée

Aucune.

Description

Méthode ; annule l'enregistrement d'un objet d'écoute d'une occurrence de composant qui diffuse un événement.

CHAPITRE 22

Composant FLVPlayback

22

Le composant FLVPlayback vous permet d'inclure facilement un lecteur vidéo à votre application Flash afin de lire des fichiers FLV (Vidéo Flash) progressivement téléchargés sur HTTP ou de lire les fichiers FLV en continu à partir de Flash Media Server (FMS) ou d'un service FVSS (Flash Video Streaming Service).

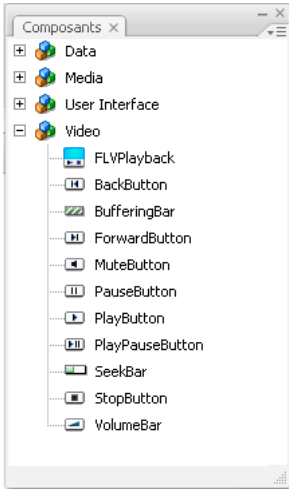
REMARQUE

Un composant FLVPlayback est pris en charge pour ActionScript 2.0 et ActionScript 3.0. Ce document présente le composant de la version 2. Si vous utilisez celui de la version 3, reportez-vous à « Utilisation du composant CheckBox » dans *Utilisation des composants ActionScript 3.0*.

Le composant FLVPlayback facile à utiliser présente les caractéristiques et les avantages suivants :

- peut être glissé sur la scène et implémenté rapidement ;
- fournit un ensemble d'*enveloppes* préconçues qui vous permettent de personnaliser l'apparence de ses contrôles de lecture ;
- permet aux utilisateurs avancés de créer leurs propres enveloppes ;
- propose des points de repère qui vous permettent de synchroniser votre vidéo avec du texte, des graphiques et de l'animation ;
- fournit un aperçu en direct des personnalisations ;
- conserve un fichier SWF de taille raisonnable.

Le composant FLVPlayback comprend les composants de l'interface utilisateur personnalisée Lecture de fichiers FLV. Il constitue une combinaison de la zone d'affichage, ou lecteur vidéo, dans laquelle vous affichez le fichier FLV et des commandes qui vous permettent de l'utiliser. Les composants de l'interface utilisateur personnalisée Lecture de fichiers FLV fournissent des boutons de commande et des mécanismes qui vous permettent de lire, d'arrêter, de mettre en pause et de contrôler autrement le fichier FLV. Ces commandes sont les suivantes : BackButton, BufferingBar, ForwardButton, MuteButton, PauseButton, PlayButton, PlayPauseButton, SeekBar, StopButton et VolumeBar. Le composant FLVPlayback et les commandes de l'interface utilisateur personnalisée Lecture de fichiers FLV apparaissent dans le panneau Composants comme indiqué sur l'illustration suivante :



La procédure consistant à ajouter des commandes de lecture au composant FLVPlayback est appelée *application d'une enveloppe*. Le composant FLVPlayback possède une enveloppe initiale par défaut, ClearOverPlaySeekMute.swf, qui fournit des commandes transparentes pour les fonctions de lecture, recherche et muette. Pour modifier cette enveloppe, vous pouvez effectuer les tâches suivantes :

- effectuer une sélection dans un ensemble d'enveloppes prédéfinies ;
- sélectionner des commandes particulières dans les composants de l'interface utilisateur personnalisée Lecture de fichiers FLV, puis les personnaliser ;
- créer une enveloppe personnalisée, puis l'ajouter à l'ensemble d'enveloppes prédéfinies.

Une fois que vous avez sélectionné une autre enveloppe, celle-ci devient la nouvelle enveloppe par défaut.

Pour plus d'informations sur la sélection ou la création d'une enveloppe pour le composant FLVPlayback, reportez-vous à « Personnalisation du composant FLVPlayback », à la page 542.

Le composant FLVPlayback inclut également une API ActionScript. Cette API inclut les classes FLVPlayback, VideoError et VideoPlayer. Pour plus d'informations sur ces classes, reportez-vous à « Classe FLVPlayback », à la page 559, à « Classe VideoPlayer », à la page 730 et à « Classe VideoError », à la page 723.

Utilisation du composant FLVPlayback

L'utilisation de base du composant FLVPlayback consiste à le placer sur la scène et à spécifier un fichier FLV à lire. Par ailleurs, vous pouvez définir d'autres paramètres qui régissent son comportement et décrivent le fichier FLV.

Création d'une application avec le composant FLVPlayback

Vous pouvez inclure le composant FLVPlayback à votre application des trois façons décrites ci-dessous :

- Faites glisser le composant FLVPlayback du panneau Composants sur la scène, puis indiquez la valeur du paramètre `contentPath`.
- Utilisez l'Assistant Importation vidéo pour créer le composant sur la scène, puis personnalisez-le en choisissant une enveloppe.
- Utilisez la méthode MovieClip `attachMovie()` pour créer de façon dynamique une occurrence FLVPlayback sur la scène, en supposant que le composant soit dans la bibliothèque.

Pour faire glisser le composant FLVPlayback à partir du panneau Composants :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser le composant FLVPlayback sur la scène.
3. Dans l'onglet Paramètres de l'inspecteur de composants, après avoir sélectionné le composant FLVPlayback sur la scène, recherchez la cellule value correspondant au paramètre `contentPath`, puis entrez une chaîne qui spécifie l'un des éléments suivants :
   - le chemin local d'un fichier FLV ;
   - l'URL d'un fichier FLV ;
   - l'URL d'un fichier XML qui décrit comment lire un fichier FLV.

   Pour plus d'informations sur la création d'un fichier XML pour décrire un ou plusieurs fichiers FLV, reportez-vous à « Utilisation d'un fichier SMIL », à la page 737.

4. Dans l'onglet Paramètres de l'inspecteur de composants, avec le composant FLVPlayback sélectionné sur la scène, double-cliquez sur le paramètre Skin (Enveloppe) pour ouvrir la boîte de dialogue de sélection d'une enveloppe.
5. Choisissez l'une des options suivantes :
   - Dans la liste déroulante Enveloppe, sélectionnez l'une des enveloppes prédéfinies pour associer un ensemble de commandes de lecture au composant.
   - Si vous avez créé une enveloppe personnalisée, sélectionnez URL d'enveloppe personnalisée dans la liste déroulante et entrez, dans la zone de texte URL, l'URL du fichier SWF contenant l'enveloppe.
   - Sélectionnez Aucune, puis faites glisser des composants de l'interface utilisateur personnalisée Lecture de fichiers FLV pour ajouter des commandes de lecture.

   REMARQUE

   Dans les deux premiers cas, un aperçu de l'enveloppe s'affiche dans le panneau de visualisation situé au-dessus du menu contextuel.
6. Cliquez sur OK pour fermer la boîte de dialogue Sélectionner une enveloppe.
7. Sélectionnez Contrôle > Tester l'animation pour exécuter le fichier SWF et démarrer la vidéo.

Pour utiliser l'Assistant Importation vidéo :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Choisissez Fichier > Importer > Importer de la vidéo.
3. Indiquez l'emplacement du fichier vidéo en sélectionnant l'une des options suivantes :
   - Sur l'ordinateur local :
   - Déjà déployé sur le Web, FMS ou serveur FVSS
4. Selon votre choix, entrez le chemin du fichier ou l'URL spécifiant l'emplacement du fichier vidéo, puis cliquez sur Suivant.

5. Si vous avez sélectionné un chemin de fichier, une boîte de dialogue Déploiement s'affiche ; vous pouvez alors sélectionner l'une des options répertoriées pour indiquer comment vous souhaitez déployer votre vidéo :
   - Téléchargement progressif à partir d'un serveur Web standard
   - Diffusion en continu avec le service FVSS
   - Diffusion en continu à partir de Flash Media Server
   - Incorporer la vidéo dans SWF et la diffuser dans le scénario

AVERTISSEMENT	Ne sélectionnez pas l'option Incorporer la vidéo. Le composant FLVPlayback lit uniquement la vidéo en continu externe. Cette option ne permet pas de placer un composant FLVPlayback sur la scène.

6. Cliquez sur Suivant.
7. Sélectionnez les paramètres de codage de votre choix.
8. Cliquez sur Suivant.
9. Choisissez l'une des options suivantes :
   - Dans la liste déroulante Enveloppe, sélectionnez l'une des enveloppes prédéfinies pour associer un ensemble de commandes de lecture au composant.
   - Si vous avez créé une enveloppe personnalisée pour le composant, sélectionnez URL d'enveloppe personnalisée dans la liste déroulante et entrez, dans la zone de texte URL, l'URL du fichier SWF contenant l'enveloppe.
   - Sélectionnez Aucune, puis faites glisser des composants de l'interface utilisateur personnalisée Lecture de fichiers FLV pour ajouter des commandes de lecture.

REMARQUE	Dans les deux premiers cas, un aperçu de l'enveloppe s'affiche dans le panneau de visualisation situé au-dessus du menu contextuel.

10. Cliquez sur Suivant.
11. Lisez la boîte de dialogue Terminer l'importation de vidéos pour savoir ce qui se passe ensuite, puis cliquez sur Terminer.
12. Enregistrez votre fichier FLA.
13. Sélectionnez Contrôle > Tester l'animation pour exécuter le fichier SWF et démarrer la vidéo.

Pour créer une occurrence de façon dynamique à l'aide d'ActionScript :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser le composant FLVPlayback du panneau Composants vers la Bibliothèque (Fenêtre > Bibliothèque).
3. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario. Remplacez *lecteur_installation* par le lecteur sur lequel vous avez installé Flash, puis modifiez le chemin pour refléter l'emplacement du dossier Skins (Enveloppes) de votre installation.

```
import mx.video.*;
this.attachMovie("FLVPlayback", "my_FLVPlybk", 10, {width:320,
  height:240, x:100, y:100});
my_FLVPlybk.skin = "file:///install_drive|/Program Files/Adobe/Adobe
  Flash CS3/en/Configuration/FLVPlayback Skins/ActionScript 2.0/
  ClearOverPlaySeekMute.swf"
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

   La méthode `attachMovie()` fait partie de la classe MovieClip. Vous pouvez l'utiliser pour créer une occurrence du composant FLVPlayback, car la classe FLVPlayback étend la classe MovieClip.

   REMARQUE Si vous ne définissez pas les propriétés `contentPath` et `skin`, le clip généré apparaît vide.

4. Sélectionnez Contrôle > Tester l'animation pour exécuter le fichier SWF et démarrer le fichier FLV.

Paramètres du composant FLVPlayback

Vous pouvez définir les paramètres suivants pour chaque occurrence du composant FLVPlayback dans l'inspecteur de composants ou l'inspecteur Propriétés :

autoPlay Valeur booléenne qui détermine comment lire le fichier FLV. Si elle est définie sur `true`, le composant lit le fichier FLV dès son chargement. Si elle est définie sur `false`, le composant charge la première image, puis une pause a lieu. La valeur par défaut est `true` pour le lecteur vidéo par défaut (0) et `false` pour les autres. Pour plus d'informations sur l'utilisation de plusieurs lecteurs vidéo dans une même occurrence FLVPlayback, reportez-vous à « Lecture de plusieurs fichiers FLV », à la page 539.

autoRewind Valeur booléenne qui détermine si le fichier FLV est rembobiné automatiquement à la fin de sa lecture. Si elle est définie sur `true`, le composant FLVPlayback rembobine automatiquement le fichier FLV au début lorsque la tête de lecture atteint la fin ou lorsque l'utilisateur clique sur le bouton Arrêter. Si elle est définie sur `false`, le composant arrête la lecture à la dernière image du fichier FLV et ne le rembobine pas automatiquement. La valeur par défaut est `true`.

autoSize Valeur booléenne qui, si elle est définie sur `true`, redimensionne le composant à l'exécution pour utiliser les dimensions du fichier FLV source. Ces dimensions sont codées dans le fichier FLV et diffèrent des dimensions par défaut du composant FLVPlayback. La valeur par défaut est `false`. Pour plus d'informations, voir « FLVPlayback.autoSize » à la page 580.

bufferTime Nombre de secondes pour mettre le fichier FLV en mémoire tampon avant de commencer la lecture. Ce paramètre affecte la diffusion des fichiers FLV en flux continu qui sont mis en mémoire tampon, mais pas téléchargés. S'il s'agit d'un fichier FLV téléchargé progressivement sur HTTP, il y a peu d'avantages à augmenter cette valeur, même si cela peut améliorer la qualité de visualisation d'une vidéo de grande qualité sur un ordinateur plus ancien, moins performant. La valeur par défaut est 0.1 Pour plus d'informations, reportez-vous à « FLVPlayback.bufferTime » à la page 592.

REMARQUE	Définir ce paramètre ne garantit pas le téléchargement d'une certaine partie du fichier FLV avant le début de la lecture.

contentPath Chaîne qui indique l'URL d'un fichier FLV ou XML expliquant comment lire un ou plusieurs fichiers FLV. Vous pouvez spécifier un chemin sur votre ordinateur local, un chemin HTTP ou un chemin RTMP (Real-Time Messaging Protocol). Double-cliquez sur la cellule value correspondant à ce paramètre pour ouvrir la boîte de dialogue Chemin du contenu. La valeur par défaut est une chaîne vide.

Si vous ne spécifiez pas de valeur pour le paramètre `contentPath`, rien ne se passe lorsque Flash exécute l'occurrence de composant FLVPlayback. Pour plus d'informations, voir « Définition du paramètre contentPath », à la page 529.

cuePoints Chaîne qui décrit les points de repère du fichier FLV. Les points de repère vous permettent de synchroniser des points spécifiques dans le fichier FLV avec une animation Flash, des graphiques ou du texte. La valeur par défaut est une chaîne vide. Pour plus d'informations, voir « Utilisation des points de repère », à la page 531.

isLive Valeur booléenne qui, si elle est définie sur `true`, spécifie la diffusion du fichier FLV en continu et en direct depuis FMS. Un flux en direct est par exemple une vidéo d'actualités à chaud. La valeur par défaut est `false`. Pour plus d'informations, voir « FLVPlayback.isLive » à la page 624.

maintainAspectRatio Valeur booléenne qui, si elle est définie sur `true`, redimensionne le lecteur vidéo dans le composant FLVPlayback afin de conserver les proportions du fichier FLV source ; le fichier FLV source est dimensionné pour s'adapter aux dimensions du composant FLVPlayback sur la scène. Le paramètre `autoSize` a la priorité sur ce paramètre. La valeur par défaut est `true`. Pour plus d'informations, voir « FLVPlayback.maintainAspectRatio » à la page 628.

skin Paramètre qui ouvre la boîte de dialogue Sélectionner une enveloppe dans laquelle vous pouvez choisir une enveloppe pour le composant. Initialement, la valeur par défaut correspond à une enveloppe prédéfinie, mais par la suite, il s'agit de la dernière enveloppe sélectionnée. Si vous sélectionnez `None`, l'occurrence FLVPlayback ne possède pas d'éléments de commande pour utiliser le fichier FLV. Si le paramètre `autoPlay` est défini sur `true`, le fichier FLV est lu automatiquement. Pour plus d'informations, voir « Personnalisation du composant FLVPlayback », à la page 542.

skinAutoHide Valeur booléenne qui, si elle est définie sur `true`, masque l'enveloppe lorsque la souris n'est pas sur le fichier FLV ou la zone d'enveloppe s'il s'agit d'une enveloppe externe qui n'est pas dans la zone d'affichage du fichier FLV. La valeur par défaut est `false`. Pour plus d'informations, voir « FLVPlayback.skin » à la page 695.

totalTime Nombre total de secondes indiqué avec une précision allant jusqu'aux millisecondes du fichier FLV source. La valeur par défaut est 0.

Si vous utilisez FMS ou FVSS, le composant calcule toujours la durée totale à partir du serveur.

Si vous utilisez le téléchargement progressif sur HTTP, le composant utilise ce nombre s'il est défini sur une valeur supérieure à zéro. Sinon il essaie de calculer la durée à partir de métadonnées du fichier FLV. Pour plus d'informations, voir « FLVPlayback.totalTime » à la page 709.

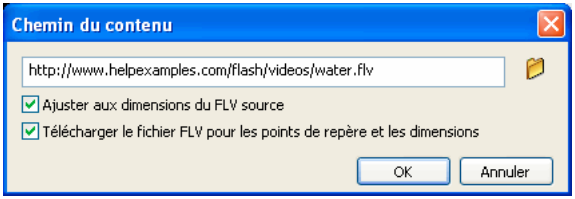
volume Nombre compris entre 0 et 100 qui représente un pourcentage du volume maximal (100). Pour plus d'informations, voir « FLVPlayback.volume » à la page 715.

Chacun de ces paramètres a une propriété équivalente dans la classe FLVPlayback. Définir la propriété remplace la définition du paramètre dans l'inspecteur de composants ou l'inspecteur Propriétés.

Définition du paramètre contentPath

Le paramètre `contentPath` permet d'indiquer le nom et l'emplacement du fichier FLV, ces deux informations indiquant à Flash comment lire le fichier.

Dans l'inspecteur de composants, ouvrez la boîte de dialogue Chemin du contenu en double-cliquant sur la cellule value correspondant au paramètre `contentPath`. La boîte de dialogue est semblable à l'illustration suivante :



La boîte de dialogue contient deux cases à cocher qui permettent de déterminer les dimensions de l'occurrence FLVPlayback et d'indiquer si vous souhaitez acquérir les dimensions et les informations sur les points de repère à partir du fichier FLV. Pour plus d'informations, voir « Options de fichier FLV », à la page 530.

Chemin du contenu

Entrez l'URL ou le chemin local du fichier FLV ou d'un fichier XML qui décrit comment lire le fichier FLV. Si vous ne connaissez pas l'emplacement exact d'un fichier FLV, cliquez sur l'icône de dossier pour ouvrir une boîte de dialogue Navigateur afin de vous aider à le trouver. Lorsque vous recherchez un fichier FLV, s'il se trouve à l'emplacement du fichier SWF cible (ou au-dessous), Flash utilise automatiquement le chemin relatif de cet emplacement afin que vous puissiez l'utiliser à partir d'un serveur Web. Autrement, il s'agit d'un chemin absolu, Windows ou Macintosh. Pour indiquer le nom d'un fichier XML local, vous devez taper son chemin et son nom.

Si vous spécifiez une URL HTTP, le fichier FLV est lu à mesure de son téléchargement progressif. Si vous spécifiez une URL RTMP, le fichier FLV est diffusé en continu à partir d'un FMS ou d'un FVSS. L'URL d'un fichier XML peut également être un fichier FLV à diffusion en flux continu à partir d'un FMS ou d'un FVSS.

ATTENTION

Si vous cliquez sur OK dans la boîte de dialogue Chemin du contenu, le composant met à jour la valeur du paramètre `cuePoints`, car il ne peut plus s'appliquer si le chemin du contenu est modifié. Par conséquent, vous *pouvez* perdre les points de repère désactivés, sauf s'il s'agit de points de repère ActionScript. (Vous ne perdez pas les points de repère désactivés si le nouveau fichier FLV contient ces mêmes points, ce qui peut se produire si vous changez simplement de chemin.) Pour cette raison, vous pouvez désactiver les points de repère non ActionScript via ActionScript plutôt qu'au moyen de la boîte de dialogue Points de repère.

Vous pouvez également spécifier l'emplacement d'un fichier XML qui décrit comment lire plusieurs flux continus de fichier FLV pour plusieurs bandes passantes. Le fichier XML utilise le langage SMIL (Synchronized Multimedia Integration Language) pour décrire les fichiers FLV. Pour obtenir une description du fichier SMIL XML, reportez-vous à « Utilisation d'un fichier SMIL », à la page 737.

Vous pouvez également indiquer le nom et l'emplacement du fichier FLV à l'aide de la propriété `FLVPlayback.contentPath` et des méthodes `FLVPlayback.play()` et `FLVPlayback.load()` ActionScript. Ces trois solutions sont prioritaires par rapport au paramètre `contentPath` de l'inspecteur de composants. Pour plus d'informations, reportez-vous à « FLVPlayback.contentPath » à la page 601, à « FLVPlayback.play() » à la page 643 et à « FLVPlayback.load() » à la page 626.

Options de fichier FLV

La boîte de dialogue Chemin du contenu contient également deux options. La première, Correspondre aux dimensions FLV source, indique si l'occurrence FLVPlayback sur la scène doit correspondre aux dimensions du fichier FLV source. Le fichier FLV source contient la hauteur et la largeur souhaitées pour la lecture. Si vous activez cette première case à cocher, l'occurrence FLVPlayback est redimensionnée pour correspondre à ces dimensions souhaitées. Cette option est cependant disponible uniquement si la seconde case à cocher est également activée.

La seconde option, Télécharger les points de repère et dimensions FLV, est activée uniquement si le chemin du contenu est une URL HTTP ou RTMP, en d'autres termes si le fichier FLV n'est pas local. Tout chemin qui ne se termine pas par .flv est également considéré comme un chemin réseau, car il peut s'agir d'un fichier XML et il peut pointer vers des fichiers FLV situés n'importe où. Cette option détermine si vous souhaitez télécharger ou diffuser en flux continu une partie du fichier FLV pour acquérir les dimensions et les définitions des points de repère intégrées dans ce fichier FLV. Flash utilise les dimensions pour redimensionner l'occurrence FLVPlayback et charge les définitions de points de repère dans le paramètre `cuePoints` dans l'inspecteur de composants. Si cette case à cocher n'est pas activée, la première ne l'est pas non plus.

Utilisation des points de repère

Un point de repère est un point au niveau duquel le lecteur vidéo distribue un événement `cuePoint` pendant la lecture d'un fichier FLV. Vous pouvez ajouter des points de repère dans un fichier FLV aux moments auxquels vous souhaitez interagir avec un autre élément de la page Web. Par exemple, vous pouvez afficher du texte ou un graphique ou bien effectuer une synchronisation avec une animation Flash ou encore modifier la lecture du fichier FLV en l'interrompant, en recherchant un autre point de repère ou en basculant vers un autre fichier FLV. Les points de repère permettent de recevoir des commandes dans le code ActionScript, et vous pouvez synchroniser ces points contenus dans votre fichier FLV avec d'autres actions de la page Web.

Il existe trois types de points de repère : de navigation, d'événement et ActionScript. Les points de repère de navigation et d'événement sont également désignés sous le nom de points de repère *intégrés*, car ils sont intégrés dans le flux continu du fichier FLV et dans le paquet de métadonnées du fichier FLV.

Un *point de repère de navigation* permet de rechercher une image particulière du fichier FLV, car il y crée une *image-clé* la plus près possible de l'heure indiquée. Une image-clé est un segment de données qui intervient entre les images du flux continu du fichier FLV. Lorsque vous recherchez un point de repère de navigation, le composant recherche cette image-clé et démarre l'événement `cuePoint`.

Un *point de repère d'événement* permet de synchroniser un point dans le temps du fichier FLV avec un événement externe de la page Web. L'événement `cuePoint` se produit exactement au moment spécifié. Vous pouvez intégrer des points de repère de navigation et d'événement dans un fichier FLV à l'aide de l'Assistant Importation vidéo ou de l'encodeur vidéo de Flash. Pour plus d'informations sur l'Assistant Importation vidéo et l'encodeur vidéo de Flash, consultez la section « Utilisation de la vidéo » dans *Utilisation de Flash*.

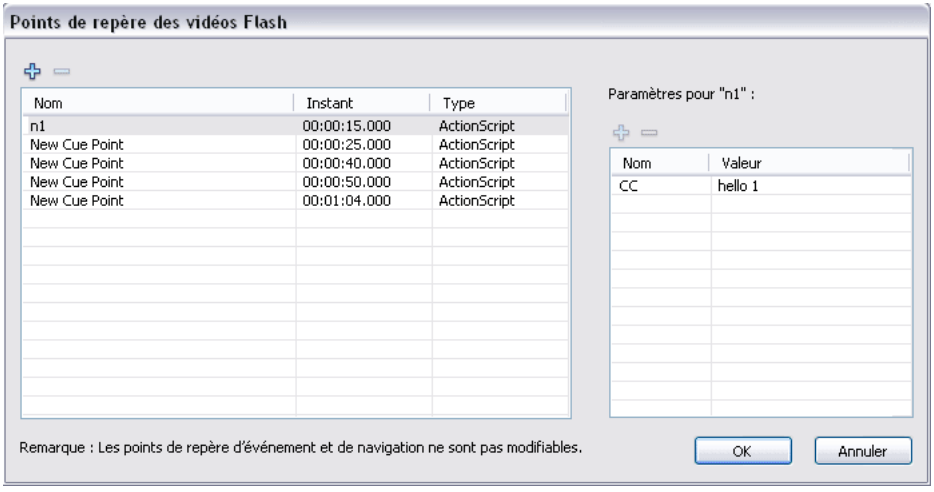
Un *point de repère ActionScript* est un point de repère externe que vous pouvez ajouter dans la boîte de dialogue Points de repère des vidéos Flash du composant ou via la méthode `FLVPlayback.addASCuePoint()`. Le composant stocke et suit les points de repère ActionScript séparément du fichier FLV. Ils sont par conséquent moins précis que les points de repère intégrés. La précision des points de repère ActionScript est d'un dixième de seconde. Vous pouvez améliorer la précision des points de repère ActionScript en diminuant la valeur de la propriété `playheadUpdateInterval`, car le composant génère l'événement `cuePoint` pour les points de repère ActionScript lors des mises à jour de la tête de lecture. Pour plus d'informations, voir « FLVPlayback.playheadUpdateInterval », à la page 649.

Dans ActionScript et dans les métadonnées du fichier FLV, un point de repère est représenté sous la forme d'un objet pourvu des propriétés suivantes : `name`, `time`, `type` et `parameters`. La propriété `name` est une chaîne qui contient le nom affecté au point de repère. La propriété `time` est un nombre qui représente l'heure exprimée en heures, minutes, secondes et millisecondes (HH:MM:SS.mmm) à laquelle se présente le point de repère. La propriété `type` est une chaîne dont la valeur est « `navigation` », « `event` » ou « `actionscript` », en fonction du type de point de repère que vous avez créé. La propriété `parameters` est un tableau de paires nom/valeur spécifiées.

Lorsqu'un événement `cuePoint` se produit, l'objet point de repère est disponible dans l'objet événement par le biais de la propriété `info`. Pour plus d'informations, voir « Ecoute des événements cuePoint », à la page 535.

Utilisation de la boîte de dialogue Points de repère des vidéos Flash

Ouvrez la boîte de dialogue Points de repère des vidéos Flash en double-cliquant sur la cellule value du paramètre `cuePoints` dans l'inspecteur de composants. La boîte de dialogue est semblable à l'illustration suivante :



La boîte de dialogue affiche des points de repère intégrés et ActionScript. Vous pouvez l'utiliser pour ajouter et supprimer des points de repère ActionScript et des paramètres cuePoints. Vous pouvez également activer ou désactiver des points de repère intégrés, mais vous n'êtes pas en mesure d'en ajouter, de les modifier ou de les supprimer.

Pour ajouter un point de repère ActionScript :

1. Sélectionnez une occurrence du composant FLVPlayback.
2. Dans l'inspecteur de composants, double-cliquez sur la cellule value du paramètre `cuePoints` pour ouvrir la boîte de dialogue Points de repère Flash.
3. Cliquez sur le signe plus (+) dans le coin supérieur gauche, au-dessus de la liste des points de repère, pour ajouter une entrée de point de repère ActionScript par défaut.
4. Cliquez sur le texte Nouveau point de repère dans la colonne Nom, puis ajoutez du texte pour nommer le point de repère.
5. Cliquez sur la valeur horaire 00:00:00:000 pour la modifier, puis affectez l'heure du point de repère. Vous pouvez spécifier l'heure en heures, minutes, secondes et millisecondes (HH:MM:SS.mmm).

   Si plusieurs points de repère existent, la boîte de dialogue place ce nouveau point à sa position chronologique dans la liste.
6. Pour ajouter un paramètre au point de repère sélectionné, cliquez sur le signe plus (+) au-dessus de la section Paramètres, puis entrez des valeurs dans les colonnes Nom et Valeur. Répétez cette étape pour chaque paramètre.
7. Pour ajouter d'autres points de repère ActionScript, répétez les étapes ci-dessus pour chaque ajout.
8. Cliquez sur OK pour enregistrer les changements.

Pour supprimer un point de repère ActionScript :

1. Sélectionnez une occurrence du composant FLVPlayback.
2. Dans l'inspecteur de composants, double-cliquez sur la cellule value du paramètre `cuePoints` pour ouvrir la boîte de dialogue Points de repère Flash.
3. Sélectionnez le point de repère à supprimer.
4. Cliquez sur le signe moins (-) dans le coin supérieur gauche, au-dessus de la liste des points de repère, pour le supprimer.
5. Répétez les étapes ci-dessus pour chaque point de repère à supprimer.
6. Cliquez sur OK pour enregistrer les changements.

Pour activer ou désactiver un point de repère intégré à un fichier FLV :

1. Sélectionnez une occurrence du composant FLVPlayback.
2. Dans l'inspecteur de composants, double-cliquez sur la cellule value du paramètre `cuePoints` pour ouvrir la boîte de dialogue Points de repère Flash.
3. Sélectionnez le point de repère à activer ou à désactiver.

4. Dans la colonne Type, cliquez sur la valeur pour déclencher le menu contextuel ou cliquez sur la flèche vers le bas.
5. Cliquez sur le nom du type de point de repère (par exemple, Evénement ou Navigation) pour l'activer. Cliquez sur Désactivé pour le désactiver.
6. Cliquez sur OK pour enregistrer les changements.

Utilisation d'ActionScript avec des points de repère

Vous pouvez utiliser ActionScript pour ajouter des points de repère ActionScript, écouter des événements cuePoint, rechercher des points de repère d'un type quelconque ou particulier, chercher un point de repère de navigation, activer ou désactiver un point de repère, vérifier si un point de repère est activé ou en supprimer un.

Les exemples figurant dans cette section utilisent le fichier FLV cuepoints.flv. Il contient les trois points de repère suivants :

Nom	Heure	Type
point1	00:00:00.418	Navigation
point2	00:00:07.748	Navigation
point3	00:00:16.020	Navigation

Ajout de points de repère ActionScript

Vous pouvez ajouter des points de repère ActionScript dans un fichier FLV à l'aide de la méthode `addASCuePoint()`. L'exemple suivant ajoute deux points de repère ActionScript dans le fichier FLV lorsqu'il est prêt. Il ajoute le premier point de repère à l'aide d'un objet point de repère qui spécifie l'heure, le nom et le type du point dans ses propriétés. Le second appel spécifie l'heure et le nom à l'aide des paramètres `time` et `name` de la méthode.

```
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv"
var cuePt:Object = new Object(); // Création d'un objet point de repère.
cuePt.time = 2.02;
cuePt.name = "ASpt1";
cuePt.type = "actionscript";
my_FLVPlybk.addASCuePoint(cuePt);  // Ajout d'un point de repère AS.
// Ajout du second point de repère AS à l'aide des paramètres d'heure et
// de nom.
my_FLVPlybk.addASCuePoint(5, "ASpt2");
```

Pour plus d'informations, voir la section « FLVPlayback.addASCuePoint() » à la page 574.

Ecoute des événements cuePoint

L'événement `cuePoint` permet de recevoir le contrôle dans le code ActionScript lorsqu'un événement `cuePoint` se produit. Lorsque des points de repère sont trouvés dans l'exemple suivant, l'écouteur `cuePoint` appelle un gestionnaire d'événements qui affiche la valeur de la propriété `playheadTime` ainsi que le nom et le type du point de repère.

```
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object):Void {
    trace("Elapsed time in seconds: " + my_FLVPlybk.playheadTime);
    trace("Cue point name is: " + eventObject.info.name);
    trace("Cue point type is: " + eventObject.info.type);
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
```

Pour plus d'informations sur l'événement cuePoint, reportez-vous à « FLVPlayback.cuePoint » à la page 602.

Recherche de points de repère

En utilisant ActionScript, vous pouvez rechercher un point de repère de tout type, le point de repère le plus proche d'une heure ou un point ayant un nom particulier.

Dans l'exemple suivant, le gestionnaire d'événements `ready` appelle la méthode `findCuePoint()` pour rechercher le point de repère `ASpt1`, puis la méthode `findNearestCuePoint()` pour rechercher le point de repère de navigation le plus proche de l'heure du point `ASpt1` :

```
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv"
var rtn_obj:Object = new Object(); // Création d'un objet point de repère.
my_FLVPlybk.addASCuePoint(2.02, "ASpt1");  // Ajout d'un point de repère
                                           // AS.
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  rtn_obj = my_FLVPlybk.findCuePoint("ASpt1");
  traceit(rtn_obj);
  rtn_obj = my_FLVPlybk.findNearestCuePoint(rtn_obj.time,
  FLVPlayback.NAVIGATION);
  traceit(rtn_obj);
}
my_FLVPlybk.addEventListener("ready", listenerObject);
function traceit(cuePoint:Object):Void {
  trace("Cue point name is: " + cuePoint.name);
  trace("Cue point time is: " + cuePoint.time);
  trace("Cue point type is: " + cuePoint.type);
}
```

Dans l'exemple suivant, le gestionnaire d'événements `ready` recherche le point de repère `ASpt` et appelle la méthode `findNextCuePointWithName()` pour rechercher le point de repère suivant portant le même nom :

```
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv"
var rtn_obj:Object = new Object(); // Création d'un objet point de repère.
my_FLVPlybk.addASCuePoint(2.02, "ASpt");  // Ajout d'un point de repère AS.
my_FLVPlybk.addASCuePoint(3.4, "ASpt");  // Ajout d'un second point de
                                         // repère AS.
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  rtn_obj = my_FLVPlybk.findCuePoint("ASpt");
  traceit(rtn_obj);
  rtn_obj = my_FLVPlybk.findNextCuePointWithName(rtn_obj);
  traceit(rtn_obj);
}
my_FLVPlybk.addEventListener("ready", listenerObject);
function traceit(cuePoint:Object):Void {
  trace("Cue point name is: " + cuePoint.name);
  trace("Cue point time is: " + cuePoint.time);
  trace("Cue point type is: " + cuePoint.type);
}
```

Pour plus d'informations sur la recherche de points de repère, reportez-vous à « FLVPlayback.findCuePoint() » à la page 608, à « FLVPlayback.findNearestCuePoint() » à la page 611 et à « FLVPlayback.findNextCuePointWithName() » à la page 614.

Recherche de points de repère de navigation

Vous pouvez rechercher un point de repère de navigation et le point de repère de navigation suivant ou précédent à une heure spécifiée. L'exemple suivant lit le fichier FLV cuepoints.flv et recherche le point de repère situé à 7.748 lorsque l'événement `ready` a lieu. Lorsque l'événement cuePoint se produit, l'exemple appelle la méthode `seekToPrevNavCuePoint()` pour rechercher le premier point de repère. Lorsque cet événement se produit, l'exemple appelle la méthode `seekToNextNavCuePoint()` pour rechercher le dernier point de repère en ajoutant 10 secondes à `eventObject.info.time`, qui correspond à l'heure du point de repère actuel.

```
import mx.video.*;

var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object) {
  my_FLVPlybk.seekToNavCuePoint("point2");
}
my_FLVPlybk.addEventListener("ready", listenerObject);
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object) {
  trace(eventObject.info.time);
  if(eventObject.info.time == 7.748)
    my_FLVPlybk.seekToPrevNavCuePoint(eventObject.info.time - .005);
  else
    my_FLVPlybk.seekToNextNavCuePoint(eventObject.info.time + 10);
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
my_FLVPlybk.contentPath = "http://helpexamples.com/flash/video/
  cuepoints.flv";
```

Pour plus d'informations, voir « FLVPlayback.seekToNavCuePoint() » à la page 685, « FLVPlayback.seekToNextNavCuePoint() » à la page 687 et « FLVPlayback.seekToPrevNavCuePoint() » à la page 688.

Activation et désactivation des points de repère intégrés au fichier FLV

Vous pouvez activer et désactiver les points de repère intégrés au fichier FLV à l'aide de la méthode `setFLVCuePointEnabled()`. Les points de repère désactivés ne déclenchent pas d'événements `cuePoint` et n'utilisent pas les méthodes `seekToCuePoint()`, `seekToNextNavCuePoint()` ni `seekToPrevNavCuePoint()`. Toutefois, vous pouvez rechercher des points de repère désactivés à l'aide des méthodes `findCuePoint()`, `findNearestCuePoint()` et `findNextCuePointWithName()`.

Vous pouvez vérifier si un point de repère intégré au fichier FLV est activé en utilisant la méthode `isFLVCuePointEnabled()`. L'exemple suivant désactive les points de repère intégrés `point2` et `point3` lorsque la vidéo est prête à être lue. Cependant, lorsque le premier événement `cuePoint` se produit, le gestionnaire d'événements vérifie si le point de repère `point3` est désactivé. Le cas échéant, il l'active.

```
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  my_FLVPlybk.setFLVCuePointEnabled(false, "point2");
  my_FLVPlybk.setFLVCuePointEnabled(false, "point3");
}
my_FLVPlybk.addEventListener("ready", listenerObject);
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object):Void {
    trace("Cue point time is: " + eventObject.info.time);
    trace("Cue point name is: " + eventObject.info.name);
    trace("Cue point type is: " + eventObject.info.type);
    if (my_FLVPlybk.isFLVCuePointEnabled("point2") == false) {
      my_FLVPlybk.setFLVCuePointEnabled(true, "point2");
    }
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
```

Pour plus d'informations, reportez-vous à « FLVPlayback.isFLVCuePointEnabled() » à la page 622 et à « FLVPlayback.setFLVCuePointEnabled() » à la page 691.

Suppression d'un point de repère ActionScript

La méthode `removeASCuePoint()` permet de supprimer un point de repère ActionScript. L'exemple suivant supprime le point de repère `ASpt2` lorsque le point de repère `ASpt1` se présente :

```
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object):Void {
    trace("Cue point name is: " + eventObject.info.name);
    if (eventObject.info.name == "ASpt1") {
      my_FLVPlybk.removeASCuePoint("ASpt2");
      trace("Removed cue point ASpt2");
    }
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
```

Pour plus d'informations, voir la section « FLVPlayback.removeASCuePoint() » à la page 661.

Lecture de plusieurs fichiers FLV

Vous pouvez lire des fichiers FLV de façon séquentielle dans une occurrence FLVPlayback en chargeant tout simplement une nouvelle URL dans la propriété `contentPath` à la fin de la lecture du fichier FLV précédent. Par exemple, le code ActionScript suivant écoute l'événement `complete` qui se produit à la fin de la lecture d'un fichier FLV. Lorsque cet événement se produit, le code définit le nom et l'emplacement d'un nouveau fichier FLV dans la propriété `contentPath`, puis appelle la méthode `play()` pour lire la nouvelle vidéo.

```
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  clouds.flv";
var listenerObject:Object = new Object();
// Ecoute de l'événement complete ; lecture d'un nouveau fichier FLV.
listenerObject.complete = function(eventObject:Object):Void {
  if (my_FLVPlybk.contentPath == "http://www.helpexamples.com/flash/video/
  clouds.flv") {
    my_FLVPlybk.play("http://www.helpexamples.com/flash/video/water.flv");
  }
};
my_FLVPlybk.addEventListener("complete", listenerObject);
```

Utilisation de plusieurs lecteurs vidéo

Vous pouvez également ouvrir plusieurs lecteurs vidéo dans une seule occurrence du composant FLVPlayback pour lire plusieurs vidéos et basculer entre eux en fonction de la lecture.

Le lecteur vidéo initial est créé lorsque vous faites glisser le composant FLVPlayback jusqu'à la scène. Le composant lui affecte automatiquement le numéro 0 et il devient lecteur par défaut. Pour créer un autre lecteur vidéo, définissez simplement la propriété `activeVideoPlayerIndex` sur un nouveau numéro. En définissant la propriété `activeVideoPlayerIndex`, vous *activez* également le lecteur vidéo spécifié, lequel sera affecté par les propriétés et les méthodes de la classe FLVPlayback. Toutefois la définition de la propriété `activeVideoPlayerIndex` ne permet pas d'afficher le lecteur vidéo. Pour le rendre *visible*, définissez la propriété `visibleVideoPlayerIndex` sur le numéro du lecteur vidéo. Pour plus d'informations sur l'interaction de ces propriétés avec les méthodes et les propriétés de la classe FLVPlayback, reportez-vous à « FLVPlayback.activeVideoPlayerIndex » à la page 571 et à « FLVPlayback.visibleVideoPlayerIndex » à la page 713.

Le code ActionScript suivant charge la propriété `contentPath` pour lire un fichier FLV dans le lecteur vidéo par défaut, puis lui ajoute un point de repère. Lorsque l'événement `ready` se produit, le gestionnaire d'événements ouvre un second lecteur vidéo en définissant la propriété `activeVideoPlayerIndex` sur le numéro 1. Il indique un fichier FLV et un point de repère pour ce second lecteur, puis rend de nouveau le lecteur par défaut (0) actif.

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
// Ajout d'un point de repère au lecteur par défaut.
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  clouds.flv";
my_FLVPlybk.addASCuePoint(3, "1st_switch");
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  // Ajout d'un second lecteur vidéo et création d'un point de repère
  // sur celui-ci.
  my_FLVPlybk.activeVideoPlayerIndex = 1;
  my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
  my_FLVPlybk.addASCuePoint(3, "2nd_switch");
  my_FLVPlybk.activeVideoPlayerIndex = 0;
};
my_FLVPlybk.addEventListener("ready", listenerObject);
```

Pour basculer vers un autre fichier FLV lors de la lecture d'un premier fichier, vous devez obtenir le contrôle pour passer dans le code ActionScript. Les points de repère vous permettent d'intervenir à des points spécifiques du fichier FLV à l'aide d'un événement `cuePoint`. Le code suivant crée un écouteur pour l'événement `cuePoint` et appelle une fonction de gestionnaire qui met en pause le lecteur vidéo actif (0), bascule vers le second lecteur (1), puis lit son fichier FLV :

```
// Création d'un objet écouteur.
var listenerObject:Object = new Object();
// Ajout d'une fonction de gestionnaire pour l'événement cuePoint.
listenerObject.cuePoint = function(eventObject:Object):Void {
     // Affichage du numéro du lecteur vidéo provoquant l'événement.
     trace("Hit cuePoint event for player: " + eventObject.vp);
     // Test du lecteur vidéo et basculement entre les fichiers FLV.
     if (eventObject.vp == 0) {
       my_FLVPlybk.pause(); // Pause du premier fichier FLV.
       my_FLVPlybk.activeVideoPlayerIndex = 1; // Activation du second
                                               // lecteur.
       my_FLVPlybk.visibleVideoPlayerIndex = 1; // Affichage du second
                                                // lecteur.
```

```
        my_FLVPlybk.play(); // Début de la lecture du nouveau
                            // lecteur/fichier FLV.
    } else if (eventObject.vp == 1) {
        my_FLVPlybk.pause(); // Pause du second fichier FLV.
        my_FLVPlybk.activeVideoPlayerIndex = 0; // Activation du premier
                                                // lecteur.
        my_FLVPlybk.visibleVideoPlayerIndex = 0; // Affichage du premier
                                                 // lecteur.
        my_FLVPlybk.play(); // Début de la lecture du premier lecteur.
    }
}
// Ajout d'un écouteur pour un événement cuePoint.
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
listenerObject.complete = function(eventObject:Object):Void {
    trace("Hit complete event for player: " + eventObject.vp);
    if (eventObject.vp == 0) {
        my_FLVPlybk.activeVideoPlayerIndex = 1;
        my_FLVPlybk.visibleVideoPlayerIndex = 1;
        my_FLVPlybk.play();
    } else {
        my_FLVPlybk.closeVideoPlayer(1);
    }
}
my_FLVPlybk.addEventListener("complete", listenerObject);
```

Lorsque vous créez un lecteur vidéo, l'occurrence FLVPlayback définit ses propriétés sur les valeur du lecteur vidéo par défaut, excepté pour les propriétés `contentPath`, `totalTime` et `isLive` que cette occurrence définit toujours sur les valeurs par défaut : respectivement chaîne vide, 0 et `false`. Elle définit la propriété `autoPlay`, dont la valeur par défaut est `true` pour le lecteur vidéo par défaut, sur `false`. La propriété `cuePoints` n'a aucun effet, et n'a pas d'incidences sur un chargement ultérieur dans le lecteur vidéo par défaut.

Les méthodes et les propriétés qui contrôlent le volume, le positionnement, les dimensions, la visibilité et les commandes de l'interface utilisateur sont toujours globales, et leur comportement n'est *pas* affecté par la définition de la propriété `activeVideoPlayerIndex`. Pour plus d'informations sur ces méthodes et propriétés et sur l'effet de la définition de la propriété `activeVideoPlayerIndex`, reportez-vous à « FLVPlayback.activeVideoPlayerIndex » à la page 571. Les autres propriétés et méthodes visent le lecteur vidéo identifié par la valeur de la propriété `activeVideoPlayerIndex`.

Néanmoins, les propriétés et les méthodes qui contrôlent les dimensions *interagissent* avec la propriété `visibleVideoPlayerIndex`. Pour plus d'informations, voir « FLVPlayback.visibleVideoPlayerIndex », à la page 713.

Diffusion de fichiers FLV en continu à partir d'un FMS

Si vous utilisez un FMS pour diffuser en continu des fichiers FLV sur le composant FLVPlayback, vous devez ajouter le fichier main.asc dans votre application FLV Flash Media Server. Vous trouverez le fichier main.asc dans les exemples disponibles en ligne à l'adresse www.adobe.com/go/learn_fl_samples_fr.

Pour configurer le FMS afin de diffuser les fichiers FLV en continu :

1. Créez un dossier dans le dossier applicatif FMS, puis nommez-le par exemple **my_application**.
2. Copiez le fichier main.asc dans le dossier my_application.
3. Créez un dossier nommé **streams** dans le dossier my_application.
4. Créez un dossier nommé **_definst_** dans le dossier streams.
5. Placez vos fichiers FLV dans le dossier **_definst_**.

Pour accéder à vos fichiers FLV sur le serveur FMS, utilisez une URL du type rtmp://*mon_serveur/mon_application/flux_continu*.flv.

our plus d'informations sur l'administration du serveur FMS, notamment la configuration d'un flux en direct, consultez la documentation FMS à l'adresse suivante www.adobe.com/support/documentation/en/flashcom/. Pour lire un flux en direct avec FMS, vous devez définir la propriété FLVPlayback `isLive` sur `true`. Pour plus d'informations, voir « FLVPlayback.isLive » à la page 624.

Personnalisation du composant FLVPlayback

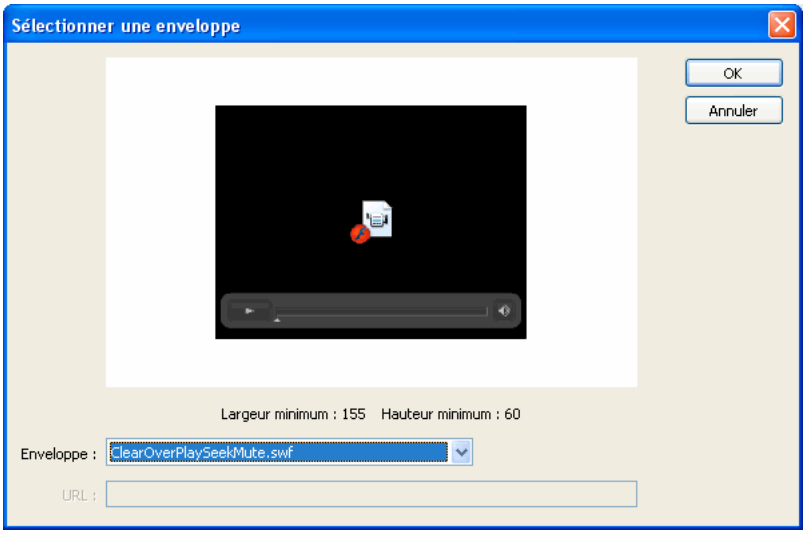
Cette section explique comment personnaliser le composant FLVPlayback. Pour une présentation complète de la personnalisation des composants comprenant la terminologie et les concepts de base d'utilisation des styles, enveloppes et thèmes, reportez-vous à *Personnalisation des composants* dans *Utilisation des composants ActionScript 2.0*. Cependant la plupart des méthodes utilisées pour personnaliser les autres composants ne fonctionnent pas avec le composant FLVPlayback. Pour le personnaliser, utilisez uniquement les techniques décrites dans cette section.

Les méthodes suivantes permettent de personnaliser le composant FLVPlayback : sélection d'une enveloppe prédéfinie, application d'une enveloppe aux composants particuliers de l'interface utilisateur personnalisée Lecture de fichiers FLV ou création d'une enveloppe. Vous pouvez également utiliser les propriétés FLVPlayback pour modifier le comportement d'une enveloppe.

REMARQUE	Vous devez transférer votre fichier SWF d'enveloppe sur le serveur Web avec votre fichier SWF d'application correspondant à l'enveloppe à utiliser avec le composant FLVPlayback.

Sélection d'une enveloppe prédéfinie

Vous pouvez choisir une enveloppe pour le composant FLVPlayback en double-cliquant sur le champ Enveloppe dans l'onglet Paramètres de l'inspecteur des composants. Dans la boîte de dialogue Sélectionner une enveloppe, vous pouvez sélectionner une enveloppe ou fournir une URL qui spécifie l'emplacement du fichier SWF d'enveloppe.



Les enveloppes qui sont répertoriées dans le menu contextuel Enveloppe sont situées dans le dossier Flash CS3 Configuration/Skins ou dans le dossier Configuration/Skins local de l'utilisateur. Vous pouvez créer des enveloppes et les faire apparaître dans cette boîte de dialogue ainsi que placer le fichier SWF dans le dossier FLVPlayback Skins/ActionScript 2.0. Le nom s'affiche dans le menu contextuel avec une extension .swf. Pour plus d'informations sur la création d'un ensemble d'enveloppes, reportez-vous à « Création d'une enveloppe », à la page 552.

Si vous souhaitez appliquer une enveloppe au composant FLVPlayback à l'aide des composants de l'interface utilisateur personnalisée Lecture de fichiers FLV, sélectionnez Aucune dans le menu contextuel.

Application d'enveloppe aux composants particuliers de l'interface utilisateur personnalisée Lecture de fichiers FLV

Les composants de l'interface utilisateur personnalisée Lecture de fichiers FLV permettent de personnaliser l'apparence des commandes FLVPlayback dans le fichier FLA et d'observer les résultats lorsque vous affichez un aperçu de votre page Web. Cependant ces composants ne sont pas conçus pour être dimensionnés. Vous devez modifier un clip et son contenu pour qu'ils soient à une taille particulière. C'est pourquoi, il est généralement conseillé de placer le composant FLVPlayback sur la scène à la taille souhaitée, les propriétés `autoSize` et `maintainAspectRatio` étant définies sur `false`.

Pour commencer, faites glisser simplement les composants de l'interface utilisateur personnalisée Lecture de fichiers FLV souhaités à partir du panneau Composants, placez-les où vous le souhaitez sur la scène, et donnez à chacun un nom d'occurrence dans l'inspecteur Propriétés. Une fois que les composants sont sur la scène, modifiez-les comme vous le feriez avec tout autre symbole.

Après avoir ouvert les composants, vous pouvez constater que la configuration de chacun diffère légèrement de celle des autres.

Composants Button

Les composants Button ont une structure similaire. Ces boutons sont les suivants : BackButton, ForwardButton, MuteButton, PauseButton, PlayButton, PlayPauseButton et StopButton. La plupart ont un seul clip sur l'image 1 avec l'occurrence placeholder_mc. Il s'agit généralement d'une occurrence de l'état normal du bouton, mais pas nécessairement. Sur l'image 2, il y a quatre clips sur la scène pour chaque état d'affichage : normal, dessus, bas et désactivé. (A l'exécution, le composant n'accède jamais réellement à l'image 2 ; ces clips sont placés ici pour faciliter les modifications et être forcés à être chargés dans le fichier SWF sans activer la case à cocher Exporter dans la première image dans la boîte de dialogue Propriétés des symboles. Cependant, vous devez toujours sélectionner l'option Exporter pour ActionScript.)

Pour appliquer une enveloppe au bouton, vous devez simplement modifier chacun de ces clips. Vous pouvez modifier leur taille ainsi que leur apparence.

Un script ActionScript s'affiche en général sur l'image 1. Normalement, vous ne devez pas le modifier. Il arrête simplement la tête de lecture sur l'image 1 et indique quels clips utiliser pour quels états.

Boutons PlayPauseButton et MuteButton

Les boutons PlayPauseButton et MuteButton sont configurés différemment des autres ; ils possèdent une seule image avec deux calques, sans script. Cette image contient deux boutons situés l'un sur l'autre : dans le premier cas, un bouton Lire et un bouton Pause, dans le second cas, un bouton de coupure du son et un bouton de restauration du son. Pour appliquer une enveloppe aux boutons PlayPauseButton ou MuteButton, appliquez une enveloppe à chacun de ces deux boutons internes, comme décrit dans « Application d'enveloppe aux composants particuliers de l'interface utilisateur personnalisée Lecture de fichiers FLV », à la page 544 ; aucune action supplémentaire n'est requise.

Boutons BackButton et ForwardButton

La configuration des boutons BackButton et ForwardButton diffère également légèrement de celle des autres. Sur l'image 2, ils contiennent des clips supplémentaires que vous pouvez utiliser comme cadre autour d'un ou des deux boutons. Ces clips ne sont pas requis et n'ont pas de fonctionnalités spéciales ; ils sont uniquement fournis pour leur commodité. Pour les utiliser, faites-les simplement glisser sur la scène à partir du panneau Bibliothèque, puis placez-les là où vous le souhaitez. Si vous n'en voulez pas, ne les utilisez pas ou supprimez-les dans le panneau Bibliothèque.

La plupart de ces boutons, tels qu'ils sont fournis, sont basés sur un ensemble commun de clips pour que vous puissez modifier l'apparence de tous les boutons à la fois. Vous pouvez utiliser cette fonctionnalité ou remplacer ces clips communs et rendre l'apparence de chaque bouton différente.

Composant BufferingBar

Le composant BufferingBar est simple : il est constitué d'une animation qui s'affiche lorsqu'il entre en mémoire tampon, et ne nécessite pas de script ActionScript spécial pour être configuré. Par défaut, il s'agit d'une barre rayée déplacée de gauche à droite, sur laquelle est placé un masque rectangulaire pour lui donner un effet « bande zébrée », mais il n'y a rien de spécial sur cette configuration.

Bien que les barres de mise en mémoire tampon des fichiers SWF d'enveloppe utilisent l'échelle à 9 découpes, car elles doivent être mises à l'échelle à l'exécution, le composant BufferingBar de l'interface utilisateur personnalisée FLV n'utilise pas et ne *peut pas* utiliser l'échelle à 9 découpes, car il comporte des clips imbriqués. Si vous souhaitez modifier la largeur ou la longueur du composant BufferingBar, vous pouvez changer son contenu plutôt que sa dimension.

Composants SeekBar et VolumeBar

Les composants SeekBar et VolumeBar sont semblables, même si leurs fonctions sont différentes. Chacun possède des poignées, utilise les mêmes mécanismes de gestion des poignées et prend en charge les clips imbriqués qu'il contient pour suivre la progression.

Il existe de nombreux emplacements où le code ActionScript du composant FLVPlayback suppose que le point d'alignement des composants SeekBar ou VolumeBar est situé dans le coin supérieur gauche du contenu. Il est donc important de conserver cette convention. Autrement, vous risquez de rencontrer des problèmes avec les poignées et avec les clips de progression et de fond.

Bien que les barres de recherche des fichiers SWF d'enveloppe utilisent l'échelle à 9 découpes, car elles doivent être mises à l'échelle à l'exécution, le composant SeekBar de l'interface utilisateur personnalisée FLV n'utilise pas et ne *peut pas* utiliser l'échelle à 9 découpes, car il comporte des clips imbriqués. Si vous souhaitez modifier la largeur ou la longueur du composant SeekBar, vous pouvez changer son contenu plutôt que sa dimension.

Poignée

Une occurrence du clip de poignée est située sur l'image 2. A l'instar des composants BackButton et ForwardButton, le composant n'accède jamais réellement à l'image 2 ; ces clips sont placés ici pour faciliter leur modification et pour être forcés à être chargés dans le fichier SWF sans activer la case à cocher Exporter dans la première image de la boîte de dialogue Propriétés des symboles. Cependant, vous devez toujours sélectionner l'option Exporter pour ActionScript.

Vous pouvez remarquer que le clip de poignée est pourvu d'un rectangle à l'arrière-plan, la valeur alpha étant définie sur 0. Ce rectangle augmente la taille de la zone active de la poignée, ce qui facilite sa préhension sans changer son apparence, de la même façon que l'état actif d'un bouton. Comme la poignée est créée de façon dynamique à l'exécution, il doit s'agir d'un clip et non d'un bouton. Ce rectangle dont la valeur alpha est définie sur 0 n'est pas nécessaire et, en règle générale, vous pouvez remplacer l'intérieur de la poignée par l'image de votre choix. Cependant, il fonctionne mieux pour conserver le point d'alignement centré horizontalement au milieu du clip de poignée.

Le code ActionScript suivant est inséré dans l'image 1 du composant SeekBar afin de gérer la poignée :

```
stop();
handleLinkageID = "SeekBarHandle";
handleLeftMargin = 2;
handleRightMargin = 2;
handleY = 11;
```

Appeler la fonction `stop()` est nécessaire en raison du contenu de l'image 2.

La deuxième ligne indique le symbole à utiliser comme poignée, et normalement vous n'avez pas besoin de la modifier si vous modifiez simplement l'occurrence du clip de poignée sur l'image 2. A l'exécution, le composant FLVPlayback crée une occurrence du clip spécifié sur la scène comme sœur de l'occurrence du composant Bar, ce qui signifie qu'elles ont le même clip parent. Ainsi, si votre barre se situe au niveau racine, votre poignée doit également se situer à ce niveau.

La variable `handleLeftMargin` détermine l'emplacement d'origine de la poignée (0 %) alors que la variable `handleRightMargin` détermine son emplacement final (100 %). Ces nombres indiquent les décalages par rapport aux extrémités gauche et droite de la barre : des nombres positifs marquent les limites à l'intérieur de la barre alors que des nombres négatifs marquent les limites à l'extérieur. Ces décalages indiquent où peut aller la poignée, en fonction de son point d'alignement. Si vous placez le point d'alignement au milieu de la poignée, les extrémités gauche et droite de celle-ci dépasseront les marges. Le point d'alignement d'un clip de barre de recherche doit être situé dans le coin supérieur gauche de son contenu pour fonctionner correctement.

La variable `handleY` détermine la position `y` de la poignée par rapport à l'occurrence de la barre. Elle est fonction des points d'alignement de chaque clip. Dans l'exemple de poignée, le point d'alignement est situé au niveau du sommet du triangle pour la placer par rapport à la partie visible, ignorant le rectangle d'état actif invisible. Le clip de barre doit donc conserver son point d'alignement dans le coin supérieur gauche de son contenu pour fonctionner correctement.

Par exemple, si un contrôle de barre est défini sur (100, 100), avec une largeur de 100 pixels, la poignée peut aller de 102 à 198 horizontalement et rester à 111 verticalement. Si vous modifiez les variables `handleLeftMargin` et `handleRightMargin` sur -2 et la variable `handleY` sur -11, la poignée peut aller de 98 à 202 horizontalement et rester à 89 verticalement.

Clips de progression et de fond

Le composant SeekBar contient un clip de *progression* et le composant VolumeBar contient un clip de *fond*, mais en pratique, ces deux composants possèdent chacun l'un ou l'autre de ces clips, n'en possèdent aucun ou possèdent les deux. Leur structure et leur comportement sont identiques, mais ils gèrent différentes valeurs. Un clip de progression se remplit à mesure du téléchargement du fichier FLV (ce qui est utile uniquement pour un téléchargement HTTP, car il est toujours plein en cas de diffusion en flux continu à partir de FMS) et un clip de fond se remplit à mesure que la poignée se déplace de gauche à droite.

Le composant FLVPlayback recherche ces occurrences de clip en fonction d'un nom d'occurrence particulier. Ainsi votre occurrence de clip de progression doit posséder votre clip de barre comme parent et le nom d'occurrence progress_mc. L'occurrence de clip de fond doit être nommée fullness_mc.

Vous pouvez définir les clips de progression et de fond avec ou sans l'occurrence de clip fill_mc imbriquée dedans. Le clip fullness_mc du composant VolumeBar affiche la méthode *avec* le clip fill_mc, et le clip progress_mc du composant SeekBar affiche la méthode *sans* le clip fill_mc.

La méthode avec le clip fill_mc imbriqué est utile si vous souhaitez un remplissage qui ne peut pas être dimensionné sans déformer l'apparence.

Dans le clip fullness_mc du composant VolumeBar, l'occurrence du clip fill_mc imbriqué est masquée. Vous pouvez la masquer lorsque vous créez le clip ou créer un masque lors de l'exécution de façon dynamique. Si vous la masquez avec un clip, nommez cette occurrence **mask_mc**, puis configurez-la pour que fill_mc s'affiche comme si le pourcentage était de 100 %. Si vous ne masquez pas fill_mc, le masque créé de façon dynamique est rectangulaire et de même taille que fill_mc à 100 %.

Le clip fill_mc est révélé avec le masque d'une des deux façons, selon que fill_mc.slideReveal est défini sur `true` ou `false`.

Si fill_mc.slideReveal est défini sur `true`, alors fill_mc est déplacé de gauche à droite pour l'exposer à travers le masque. A 0 %, il est tout à fait à gauche, si bien que rien n'est affiché à travers le masque. A mesure que le pourcentage augmente, il se déplace vers la droite, jusqu'à ce qu'il retrouve, en atteignant 100 %, son emplacement d'origine lors de sa création sur la scène.

Si fill_mc.slideReveal est défini sur `false` ou non défini (comportement pas défaut), le masque est redimensionné de gauche à droite pour révéler davantage de fill_mc. Lorsqu'il est sur 0 %, le masque est redimensionné horizontalement à 05, et à mesure que le pourcentage augmente, `_xscale` augmente, jusqu'à ce qu'il révèle l'intégralité de fill_mc lorsqu'il est sur 100 %. Cela ne correspond pas nécessairement à `_xscale` = 100, car mask_mc peut avoir été redimensionné au moment de sa création.

La méthode sans fill_mc est plus simple que la méthode avec fill_mc, mais elle déforme le remplissage horizontalement. Si vous ne souhaitez pas de déformation, vous devez utiliser fill_mc. Le clip progress_mc du composant SeekBar illuste cette méthode.

Le clip de progression ou de fond est redimensionné horizontalement en fonction du pourcentage. A 0 %, `_xscale` de l'occurrence est défini sur 0, la rendant invisible. A mesure que le pourcentage augmente, `_xscale` est réglé jusqu'à ce que le clip ait sa taille d'origine au moment de sa création sur la scène lorsqu'il est sur 100 %. De nouveau, cela ne correspond pas nécessairement à `_xscale` = 100, car cette occurrence du masque peut avoir été redimensionnée au moment de sa création.

Connexion aux composants de l'interface utilisateur personnalisée Lecture de fichiers FLV

Vous devez écrire du code ActionScript pour connecter vos composants de l'interface utilisateur personnalisée Lecture de fichiers FLV à l'occurrence du composant FLVPlayback. Vous devez d'abord nommer l'occurrence FLVPlayback, puis utiliser ActionScript pour affecter les occurrences des composants de l'interface utilisateur personnalisée Lecture de fichiers FLV aux propriétés FLVPlayback correspondantes. Dans l'exemple suivant, l'occurrence FLVPlayback est nommée my_FLVPlybk, les noms des propriétés FLVPlayback suivent les points (.), et les occurrences des commandes de l'interface utilisateur personnalisée Lecture de fichiers FLV figurent à droite du signe égal (=) :

```
// Occurrence FLVPlayback = my_FLVPlybk.
my_FLVPlybk.playButton = playbtn; // Définition de la propriété playButton
                                  // sur playbtn, etc.
my_FLVPlybk.pauseButton = pausebtn;
my_FLVPlybk.playPauseButton = playpausebtn;
my_FLVPlybk.stopButton = stopbtn;
my_FLVPlybk.muteButton = mutebtn;
my_FLVPlybk.backButton = backbtn;
my_FLVPlybk.forwardButton = forbtn;
my_FLVPlybk.volumeBar = volbar;
my_FLVPlybk.seekBar = seekbar;
my_FLVPlybk.bufferingBar = bufbar;
```

Exemple

Les procédures suivantes permettent de créer les commandes StopButton, PlayPauseButton, MuteButton et SeekBar personnalisées.

1. Faites glisser le composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**.
2. A l'aide de l'inspecteur de composants, définissez le paramètre `contentPath` sur **http://www.helpexamples.com/flash/video/cuepoints.flv**.
3. Définissez le paramètre Skin sur None.
4. Faites glisser un composant StopButton, PlayPauseButton et MuteButton sur la scène, puis placez-les au-dessus de l'occurrence FLVPlayback, en les empilant verticalement sur la gauche. Donnez à chaque bouton un nom d'occurrence dans l'inspecteur des propriétés (par exemple **my_stopbttn**, **my_plypausbttn** et **my_mutebttn**).
5. Dans le panneau Bibliothèque, ouvrez le dossier Skins de FLVPlayback, puis son sous-dossier SquareButton.
6. Sélectionnez le clip SquareBgDown, puis double-cliquez dessus pour l'ouvrir sur la scène.
7. Cliquez avec le bouton droit (Windows) ou en appuyant sur la touche Contrôle (Macintosh), puis sélectionnez Sélectionner tout dans le menu, puis supprimez le symbole.
8. Sélectionnez l'outil oval, dessinez un ovale au même emplacement, puis définissez le remplissage sur bleu (#0033FF).
9. Dans l'inspecteur des propriétés, définissez la largeur (L:) sur 40 et la hauteur (H:) sur 20. Définissez la coordonnée x (X:) sur 0.0 et la coordonnée y (Y:) sur 0.0.
10. Répétez les étapes 6 à 8 pour le clip SquareBgNormal, mais changez le remplissage en le définissant sur jaune (#FFFF00).
11. Répétez les étapes 6 à 8 pour le clip SquareBgOver, mais changez le remplissage en le définissant sur vert (#006600).
12. Modifiez les clips pour les différentes icônes de symboles dans les boutons (PauseIcon, PlayIcon, MuteOnIcon, MuteOffIcon et StopIcon). Vous pouvez trouver ces clips dans le panneau Bibliothèque sous FLV Playback Skins/*Etiquette* Button/Assets, où *Etiquette* correspond au nom du bouton, par exemple Lire, Pause, etc. Exécutez la procédure pour chacun :
    a. Cliquez sur l'option Sélectionner tout.
    b. Modifiez la couleur en la définissant sur rouge (#FF0000).
    c. Redimensionnez à 300 %.

d. Modifiez l'emplacement X: du contenu sur 7.0 pour changer le placement horizontal de l'icône dans tous les états de bouton.

REMARQUE	En modifiant ainsi l'emplacement, vous n'avez pas à ouvrir chaque état de bouton et à déplacer l'occurrence de clip d'icône.

13. Cliquez sur la flèche Retour en bleu située au-dessus du scénario pour revenir à l'image 1 de la séquence 1.
14. Faites glisser un composant SeekBar sur la scène, puis placez-le dans le coin inférieur droit de l'occurrence FLVPlayback.
15. Dans le panneau Bibliothèque, double-cliquez sur le composant SeekBar pour l'ouvrir sur la scène.
16. Redimensionnez-le à 400 %.
17. Sélectionnez le contour, puis définissez la couleur sur rouge (#FF0000).
18. Double-cliquez sur le composant SeekBarProgress dans le dossier FLVPlayback Skins/Seek Bar, puis définissez la couleur sur jaune (#FFFF00).
19. Double-cliquez sur le composant SeekBarHandle dans le dossier FLVPlayback Skins/Seek Bar, puis définissez la couleur sur rouge (#FF0000).
20. Cliquez sur la flèche Retour en bleu située au-dessus du scénario pour revenir à l'image 1 de la séquence 1.
21. Sélectionnez l'occurrence SeekBar sur la scène et nommez-la **my_seekbar**.
22. Dans le panneau Actions situé sur l'image 1 du scénario, ajoutez une instruction import pour les classes Video, puis affectez les noms de bouton et de barre de recherche aux propriétés FLVPlayback correspondantes, comme indiqué dans l'exemple suivant :

```
import mx.video.*;
my_FLVPlybk.stopButton = my_stopbttn;
my_FLVPlybk.playPausebutton = my_plypausbttn;
my_FLVPlybk.muteButton = my_mutebttn;
my_FLVPlybk.seekBar = my_seekbar;
```

23. Appuyez sur Ctrl+Entrée pour tester l'animation.

Création d'une enveloppe

La meilleure façon de créer un fichier SWF d'enveloppe est de copier l'un des fichiers d'enveloppe fournis avec Flash, puis de l'utiliser comme point de départ. Vous pouvez trouver les fichiers FLA correspondant à ces enveloppes dans le dossier applicatif Flash CS3 Configuration/FLVPlayback Skins/FLA/ActionScript 2.0. Pour transformer votre fichier SWF d'enveloppe terminé en option dans la boîte de dialogue Sélectionner une enveloppe, placez-le dans le dossier Configuration/FLVPlayback Skins/ActionScript 2.0 du dossier de l'application Flash ou dans le dossier Configuration/FLVPlayback Skins/ActionScript 2.0 local d'un utilisateur.

Vous allez constater qu'il est assez facile de changer l'apparence d'un bouton ou de son *chrome* (arrière-plan) sans en modifier les dimensions. Toutes les enveloppes installées ont les mêmes boutons basés sur des chromes de différentes couleurs. Vous pouvez donc apporter des modifications significatives en changeant simplement la couleur du chrome. Vous pouvez apporter d'autres modifications, par exemple réorganiser les commandes du clip de disposition, en déplaçant simplement les clips de balise d'emplacement. Vous pouvez observer ces modifications exactement telles qu'elles apparaîtront dans le fichier SWF terminé.

Lorsque vous consultez les fichiers FLA d'enveloppe installés, vous pouvez penser que certains éléments de la scène sont superflus, mais nombre de ces éléments sont insérés dans les calques-guides. Pour voir rapidement ce qui apparaît réellement dans le fichier SWF, sélectionnez Contrôle > Tester l'animation pour effectuer une vérification du clip. Vous pouvez voir ainsi dans quelle mesure l'échelle à 9 découpes affecte certaines commandes, car elle n'est pas effective lors de la création.

Les sections ci-après présentent des personnalisations et des modifications plus complexes des clips SeekBar, BufferingBar et VolumeBar.

Utilisation du clip layout_mc

Lorsque vous ouvrez un fichier FLA d'enveloppe, un clip nommé layout_mc se trouve dans le coin supérieur gauche de la scène. Ce clip *doit* être nommé `l`ayout_mc. Le clip layout_mc et le code ActionScript qui figurent sur la même image définissent la disposition des commandes à l'exécution.

Même si le clip layout_mc ressemble beaucoup à l'apparence de l'enveloppe à l'exécution, son contenu n'est pas visible à l'exécution. Il est utilisé uniquement pour calculer l'emplacement des commandes. Les autres commandes sur la scène sont utilisées à l'exécution.

Le clip layout_mc contient une balise d'emplacement pour le composant FLVPlayback nommé video_mc. Toutes les autres commandes sont disposées par rapport à video_mc. Si vous commencez par l'un des fichiers FLA et que vous modifiez la taille des commandes, vous pouvez probablement corriger la disposition en déplaçant ces clips de balise d'emplacement.

Chacun de ces clips de balise d'emplacement a un nom d'occurrence particulier. Il s'agit des noms suivants : playpause_mc, play_mc, pause_mc, stop_mc, back_mc, bufferingBar_mc, seekBar_mc, volumeMute_mc et volumeBar_mc.

Le type de clip utilisé pour une commande n'est pas important. En règle générale, le clip d'état normal est utilisé pour les boutons. S'il s'agit des autres commandes, le clip correspondant à la commande en question est utilisé, mais c'est uniquement pour une raison de commodité. L'emplacement des coordonnées *x* (axe horizontal) et *y* (axe vertical) ainsi que la hauteur et la largeur de la balise d'emplacement sont les seuls élements qui importent.

Outre les commandes standard, vous pouvez posséder autant de clips d'arrière-plan et de premier plan que vous le souhaitez. Vous devez cependant utiliser la convention d'appellation suivante : bg1_mc, bg2_mc, etc. pour les clips d'arrière-plan et fg1_mc, fg2_mc, etc. pour les clips de premier plan. Vous ne pouvez pas sauter de nombres. Si, par exemple, vous avez des clips bg1_mc et bg3_mc, mais pas de clip bg2_mc, bg3_mc ne sera pas utilisé. Ce modèle est conçu pour placer les clips d'arrière-plan derrière les commandes, le clip bg1_mc étant placé en bas et le clip bg2_mc au-dessus, et les clips de premier plan au-dessus des commandes, avec fg1_mc en premier, fg2_mc au-dessus de fg1_mc, etc. La relation des clips est néanmoins réellement déterminée par l'ordre des commandes correspondantes sur la scène. Veillez donc à ce que cet ordre soit correct.

Le clip bg1_mc est spécial. Si vous définissez la propriété `FlvPlayback.skinAutoHide` sur `true`, l'enveloppe s'affiche lorsque la souris est au-dessus du clip bg1_mc. Cela est important pour les enveloppes qui s'affichent hors des limites du lecteur vidéo. Pour plus d'informations sur la propriété `skinAutoHide`, reportez-vous à « Modification du comportement d'enveloppe », à la page 559.

Dans les fichiers FLA, le clip bg1_mc est utilisé pour le chrome, et certains utilisent le clip bg2_mc pour la bordure entourant les boutons Avance et Retour.

ActionScript

En règle générale, le code ActionScript suivant s'applique à toutes les commandes. Certaines commandes contiennent du code ActionScript spécifique qui définit un comportement supplémentaire ; il est expliqué dans la section correspondant à la commande en question.

Le code ActionScript initial définit les largeur et hauteur minimales de l'enveloppe. La boîte de dialogue Sélectionner une enveloppe affiche ces valeurs qui sont utilisées à l'exécution pour empêcher de redimensionner l'enveloppe en deça de sa taille minimale. Si vous ne souhaitez pas spécifier de taille minimale, laissez-les non définies ou affectez-leur une valeur inférieure ou égale à zéro.

```
// Largeur et hauteur minimales de la vidéo recommandées pour l'utilisation
// de cette enveloppe.
// Laisser ces valeurs non définies ou <= 0 si pas de minimum.
layout_mc.minWidth = 270;
layout_mc.minHeight = 60;
```

Les propriétés suivantes peuvent être appliquées à chaque balise d'emplacement :

Propriété	Description
`mc:MovieClip`	Occurrence sur la scène correspondant à cette commande. Si elle n'est pas définie, `layout_mc.foo_mc.mc` devient par défaut `foo_mc`.
`anchorLeft:Boolean`	Positionne la commande par rapport au côté gauche de l'occurrence FLVPlayback. Elle est définie par défaut sur `true` sauf si `anchorRight` est explicitement définie sur `true`, puis elle est définie par défaut sur `false`.
`anchorRight:Boolean`	Positionne la commande par rapport au côté droit de l'occurrence FLVPlayback. La valeur par défaut est `false`.
`anchorBottom:Boolean`	Positionne la commande par rapport au bas de l'occurrence FLVPlayback. Elle est définie par défaut sur `true` sauf si `anchorTop` est explicitement définie sur `true`, puis elle est définie par défaut sur `false`.
`anchorTop:Boolean`	Positionne la commande par rapport au haut de l'occurrence FLVPlayback. La valeur par défaut est `false`.

Si les deux propriétés `anchorLeft` et `anchorRight` sont définies sur `true`, la commande est redimensionnée à l'horizontale lors de l'exécution. Si les deux propriétés `anchorTop` et `anchorBottom` sont définies sur `true`, la commande est redimensionnée à la verticale lors de l'exécution.

Pour connaître les effets de ces propriétés, observez leur utilisation dans les enveloppes. Les commandes BufferingBar et SeekBar sont les seules à pouvoir être redimensionnées, et elles sont disposées l'une au-dessus de l'autre. Les propriétés `anchorLeft` et `anchorRight` de ces deux commandes sont définies sur `true`. La propriété `anchorLeft` de toutes les commandes situées à gauche de BufferingBar et de SeekBar est définie sur `true`, et la propriété `anchorRight` de toutes les commandes situées à leur droite est définie sur `true`. La propriété `anchorBottom` de toutes les commandes est définie sur `true`.

Vous pouvez modifier le clip `layout_mc` pour faire une enveloppe dans laquelle les commandes sont situées en haut plutôt qu'en bas. Pour ce faire, vous devez simplement déplacer les commandes jusqu'en haut par rapport à `video_mc`, puis définir la propriété `anchorTop` de toutes les commandes sur `true`.

Etats de bouton

Tous les états de bouton sont disposés sur la scène, mais l'emplacement de ces occurrences de clip sur la scène n'est pas important. En revanche, il importe qu'ils soient imbriqués dans des clips d'une façon particulière et que toutes les occurrences de clip soient nommées correctement.

La structure des occurrences de clip ainsi que leurs noms sont affichés dans l'exemple suivant :

```
playpause_mc
  play_mc
    up_mc, over_mc, down_mc, disabled_mc
  pause_mc
    up_mc, over_mc, down_mc, disabled_mc
stop_mc
  up_mc, over_mc, down_mc, disabled_mc
back_mc
  up_mc, over_mc, down_mc, disabled_mc
forward_mc
  up_mc, over_mc, down_mc, disabled_mc
volumeMute_mc
  on_mc
    up_mc, over_mc, down_mc, disabled_mc
  off_mc
    up_mc, over_mc, down_mc, disabled_mc
```

Notez que les fichiers FLA contiennent les boutons supplémentaires Avance et Retour sur la scène. Ils sont situés sur les calques de guide et servent à montrer l'utilisation des clips ForwardBackBorder, ForwardBorder et BackBorder. Pour plus d'informations, voir « Clips d'arrière-plan et de premier plan », à la page 558.

Vous pouvez modifier les différents états comme vous le souhaitez. N'oubliez pas que tous les états sont placés au même endroit par leurs points d'alignement. Ainsi, si certains états sont plus volumineux que d'autres, vous risquez de ne pas pouvoir positionner votre bouton à (0, 0) comme c'est le cas pour la plupart des enveloppes de bouton. Dans certains cas, vous pouvez trouver plus simple de conserver le point d'alignement au centre.

Si vous ne souhaitez pas utiliser tous les états, vous pouvez en omettre certains, mais vous devez inclure up_mc. Ce clip est utilisé pour les états omis.

Si vous souhaitez disposer de boutons Lire et Pause distincts au lieu d'un bouton combiné Lire-Pause, placez simplement les clips play_mc et pause_mc sur la scène sans les lier à un clip playpause_mc.

Outre le code décrit dans « Utilisation du clip layout_mc », à la page 552, aucun code ActionScript supplémentaire n'est nécessaire pour configurer les boutons.

Barre de mise en mémoire tampon

La barre de mise en mémoire tampon possède deux clips : bufferingBar_mc et bufferingBarFill_mc. La position de chaque clip sur la scène par rapport à l'autre clip est importante, car ce positionnement relatif est conservé. La barre de mise en mémoire tampon utilise deux clips distincts, car ce composant redimensionne bufferingBar_mc, mais pas bufferingBarFill_mc.

L'échelle à 9 découpes est appliquée au clip bufferingBar_mc. De cette façon, les bordures ne sont pas déformées lors du redimensionnement. Le clip bufferingBarFill_mc est extrêmement large ; il le sera toujours suffisamment et vous n'aurez pas besoin de le redimensionner.
A l'exécution, il est automatiquement masqué pour n'afficher que la partie située au-dessus du clip bufferingBar_mc étiré. Par défaut, les dimensions exactes du masque conservent une marge égale à gauche et à droite dans le clip bufferingBar_mc, basée sur la différence existant entre les positions *x* (axe horizontal) des clips bufferingBar_mc et bufferingBarFill_mc.
Vous pouvez personnaliser le positionnement avec du code ActionScript.

Si votre barre de mise en mémoire tampon n'a pas besoin d'être redimensionnée ou n'utilise pas l'échelle à 9 découpes, vous pouvez la configurer comme le composant BufferingBar de l'interface utilisateur personnalisée Lecture de fichiers FLV. Pour plus d'informations, voir « Composant BufferingBar », à la page 546.

La barre de mise en mémoire tampon dispose de la propriété suivante supplémentaire :

Propriété	Description
`fill_mc:MovieClip`	Indique le nom d'occurrence du remplissage de la barre de mise en mémoire tampon. Propriété par défaut de bufferingBarFill_mc.

Barre de recherche et barre de volume

La barre de recherche possède également deux clips : seekBar_mc et seekBarProgess_mc. La position de chaque clip sur la scène par rapport à l'autre clip est importante, car ce positionnement relatif est conservé. Même si les deux clips sont redimensionnés, le clip seekBarProgress_mc ne peut pas être imbriqué dans le clip seekBar_mc, car seekBar_mc utilise l'échelle à 9 découpes, qui ne fonctionne pas bien avec les clips imbriqués.

L'échelle à 9 découpes est appliquée au clip seekBar_mc. De cette façon, les bordures ne sont pas déformées lors du redimensionnement. Le clip seekBarProgress_mc peut également être redimensionné, mais il subit des déformations. Il n'utilise pas l'échelle à 9 découpes, car il s'agit d'un remplissage dont l'apparence est correcte s'il est déformé.

Le clip seekBarProgress_mc fonctionne sans fill_mc, de la même façon qu'un clip progress_mc fonctionne dans les composants de l'interface utilisateur personnalisée Lecture de fichiers FLV. En d'autres mots, il n'est pas masqué et il est redimensionné à l'horizontale. Les dimensions exactes du clip bufferingBarProgress_mc à 100 % sont définies par les marges gauche et droite du clip bufferingBar_mc. Ces dimensions sont, par défaut, égales et basées sur la différence entre les positions *x* (axe horizontal) des clips seekBar_mc et seekBarProgress_mc. Avec ActionScript, vous pouvez personnaliser les dimensions du clip de barre de recherche, comme indiqué dans l'exemple suivant :

```
progressLeftMargin = 2;
progressRightMargin = 2;
progressY = 11;
fullnessLeftMargin = 2;
fullnessRightMargin = 2;
fullnessY = 11;
```

Comme dans le composant SeekBar de l'interface utilisateur personnalisée Lecture de fichiers FLV, vous pouvez créer un clip de fond pour la barre de recherche. Si vous n'avez pas besoin de redimensionner votre barre de recherche, ou si elle est redimensionnée sans utiliser l'échelle à 9 découpes, vous pouvez configurer les clips progress_mc ou fullness_mc à l'aide des méthodes utilisées pour les composants de l'interface utilisateur personnalisée Lecture de fichiers FLV. Pour plus d'informations, voir « Clips de progression et de fond », à la page 548.

Comme la barre de volume des enveloppes ne peut pas être redimensionnée, elle est construite de la même façon que le composant VolumeBar de l'interface utilisateur personnalisée Lecture de fichiers FLV. Pour plus d'informations, voir « Composants SeekBar et VolumeBar », à la page 546. L'exception concerne la poignée dont l'implémentation est différente. Pour plus d'informations à ce sujet, consultez la section suivante.

Poignée

Les poignées SeekBar et VolumeBar sont placées sur la scène en regard de la barre. Par défaut, les marges gauche et droite et les valeurs de l'axe *y* de la poignée sont définis par sa position par rapport au clip de barre. La marge gauche est définie par la différence entre l'emplacement des *x* (axe horizontal) de la poignée et l'emplacement des *x* (axe horizontal) de la barre, la marge droite étant égale à la marge gauche. Vous pouvez personnaliser ces valeurs via ActionScript dans le clip SeekBar ou VolumeBar. L'exemple suivant applique le même code ActionScript que celui utilisé dans les composants de l'interface utilisateur personnalisée Lecture de fichiers FLV :

```
handleLeftMargin = 2;
handleRightMargin = 2;
handleY = 11;
```

Au-delà de ces propriétés, les poignées sont de simples clips, configurés de la même façon que dans les composants de l'interface utilisateur personnalisée Lecture de fichiers FLV. Les deux poignées sont dotées d'arrière-plans rectangulaires, la propriété `alpha` étant définie sur 0. Elles sont uniquement destinées à augmenter la zone active et ne sont pas obligatoires.

ActionScript

La barre de recherche et la barre de volume prennent en charge les propriétés supplémentaires suivantes :

Propriété	Description
`handle_mc:MovieClip`	Clip de la poignée. Est définie par défaut pour seekBarHandle_mc ou volumeBarHandle_mc.
`progress_mc:MovieClip`	Clip de progression. Est défini par défaut pour seekBarProgress_mc ou volumeBarProgress_mc.
`fullness_mc:MovieClip`	Clip de fond. Est défini par défaut pour seekBarFullness ou volumeBarFullness.

Clips d'arrière-plan et de premier plan

Les clips chrome_mc et forwardBackBorder_mc sont implémentés comme clips d'arrière-plan.

Parmi les clips ForwardBackBorder, ForwardBorder et BackBorder situés sur la scène et les boutons de balise d'emplacement Avance et Retour, ForwardBackBorder est le seul à ne *pas* être sur un calque de guide. Ce clip figure uniquement dans les enveloppes qui utilisent réellement les boutons Avance et Retour.

Aucun code ActionScript supplémentaire n'est requis pour configurer les clips d'arrière-plan et de premier plan.

Modification du comportement d'enveloppe

Les propriétés `bufferingBarHidesAndDisablesOthers` et `skinAutoHide` permettent de personnaliser le comportement de l'enveloppe FLVPlayback.

Si vous définissez la propriété `bufferingBarHidesAndDisablesOthers` sur `true`, le composant FLVPlayback masque SeekBar et sa poignée, et désactive les boutons Lire et Pause lorsqu'il entre dans l'état de mise en mémoire tampon. Cela peut être utile lorsqu'un fichier FLV est diffusé en flux continu à partir de FMS par le biais d'une connexion lente, la valeur de la propriété `bufferTime` étant élevée (10, par exemple). Dans cette situation, un utilisateur impatient peut essayer de lancer la recherche en cliquant sur les boutons Lire et Pause, ce qui peut retarder encore davantage la lecture du fichier. Pour empêcher cela, définissez `bufferingBarHidesAndDisablesOthers` sur `true`, puis désactivez l'élément SeekBar et les boutons Pause et Lire alors que le composant est en état de mise en mémoire tampon.

La propriété `skinAutoHide` affecte uniquement les fichiers SWF d'enveloppe prédéfinis, mais pas les commandes créées dans les composants de l'interface utilisateur personnalisée Lecture de fichiers FLV. Si cette propriété est définie sur `true`, le composant FLVPlayback masque l'enveloppe lorsque la souris n'est pas sur la zone d'affichage. La valeur par défaut de cette propriété est `true`.

Classe FLVPlayback

Héritage MovieClip > Classe FLVPlayback

Nom de classe ActionScript `mx.video.FLVPlayback`

FLVPlayback étend la classe MovieClip et englobe un objet VideoPlayer. Pour plus d'informations sur la classe VideoPlayer, reportez-vous à « Classe VideoPlayer », à la page 730.

REMARQUE	La classe FLVPlayback est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Contrairement aux autres composants, le composant FLVPlayback n'étend pas UIObject ni UIComponent. Il ne prend donc pas en charge les méthodes et les propriétés de ces classes. Par exemple, pour créer une occurrence du composant dans ActionScript, vous devez appeler la méthode MovieClip, `attachMovie()`, plutôt que la méthode UIObject, `createClassObject()`.

Les méthodes et les propriétés de la classe FLVPlayback vous permettent de lire et de manipuler des fichiers FLV en utilisant le composant FLVPlayback dans votre application Flash.

La définition d'une propriété de la classe FLVPlayback avec ActionScript écrase un paramètre équivalent qui définit initialement la propriété dans l'inspecteur des propriétés ou l'inspecteur de composants.

Toutes les classes de composants ont une propriété `version` qui correspond à une propriété de classe. Les propriétés de classe ne sont disponibles que pour la classe elle-même. La propriété `version` renvoie une chaîne qui indique la version du composant. Le code suivant affiche la version dans le panneau Sortie :

```
trace(mx.video.FLVPlayback.version);
```

Récapitulatif des méthodes de la classe FLVPlayback

Le tableau suivant répertorie les méthodes de la classe FLVPlayback.

Méthode	Description
`FLVPlayback.addASCuePoint()`	Ajoute un point de repère ActionScript.
`FLVPlayback.addEventListener()`	Crée un écouteur pour un événement spécifié.
`FLVPlayback.bringVideoPlayerToFront()`	Place un lecteur vidéo devant la pile des lecteurs vidéo.
`FLVPlayback.closeVideoPlayer()`	Ferme NetStream pour le lecteur vidéo ayant l'index spécifié, puis supprime ce lecteur vidéo.
`FLVPlayback.findCuePoint()`	Recherche le type spécifié de point de repère, dont l'heure, le nom ou l'heure et le nom sont spécifiés.
`FLVPlayback.findNearestCuePoint()`	Recherche le type spécifié de point de repère situé à ou approximativement à l'heure dite ou portant le nom donné.
`FLVPlayback.findNextCuePointWithName()`	Recherche le point de repère suivant portant le même nom que celui renvoyé par les méthodes `findCuePoint()` ou `findNearestCuePoint()`.
`FLVPlayback.getVideoPlayer()`	Obtient le lecteur vidéo spécifié par le paramètre *`index`*.
`FLVPlayback.isFLVCuePointEnabled()`	Renvoie la valeur `false` si le point de repère intégré au fichier FLV est désactivé par ActionScript.
`FLVPlayback.load()`	Commence à charger le fichier FLV, la propriété `autoPlay` étant définie sur `false`.
`FLVPlayback.pause()`	Interrompt la lecture du flux vidéo en continu.

Méthode	Description
`FLVPlayback.play()`	Commence à lire le flux vidéo en continu et permet également le chargement et la lecture d'un nouveau fichier FLV.
`FLVPlayback.removeASCuePoint()`	Supprime un point de repère ActionScript.
`FLVPlayback.removeEventListener()`	Supprime un écouteur d'événement.
`FLVPlayback.seek()`	Recherche une heure donnée (en secondes) dans le fichier, avec une précision décimale allant jusqu'aux millisecondes.
`FLVPlayback.seekPercent()`	Recherche dans un pourcentage du fichier.
`FLVPlayback.seekSeconds()`	Identique à la méthode `FLVPlayback.seek()`.
`FLVPlayback.seekToNavCuePoint()`	Recherche le point de repère de navigation avec le nom donné à l'heure (ou après l'heure) spécifiée.
`FLVPlayback.seekToNextNavCuePoint()`	Recherche le point de repère de navigation suivant, en fonction de l'heure spécifiée.
`FLVPlayback.seekToPrevNavCuePoint()`	Recherche le point de repère de navigation précédent, en fonction de l'heure spécifiée.
`FLVPlayback.setFLVCuePointEnabled()`	Active ou désactive un ou plusieurs points de repère de fichier FLV.
`FLVPlayback.setScale()`	Définit simultanément `scaleX` et `scaleY`.
`FLVPlayback.setSize()`	Définit simultanément les propriétés `width` et `height`.
`FLVPlayback.stop()`	Arrête la lecture du flux vidéo en continu.

Propriétés de la classe FLVPlayback

La classe FLVPlayback possède à la fois des propriétés de classe et d'occurrence.

Propriétés de la classe FLVPlayback

Les propriétés indiquées ci-dessous ne concernent que la classe FLVPlayback. Il s'agit de constantes en lecture seule qui s'appliquent à toutes les occurrences du composant FLVPlayback de votre application.

Propriété	Valeur	Description
`FLVPlayback.ACTIONSCRIPT`	`"actionscript"`	Peut être utilisée comme type de paramètre pour les méthodes `findCuePoint()` et `findNearestCuePoint()`.
`FLVPlayback.ALL`	`"all"`	Peut être utilisée comme type de paramètre pour les méthodes `findCuePoint()` et `findNearestCuePoint()`.
`FLVPlayback.BUFFERING`	`"buffering"`	Valeur permettant de tester la propriété d'état.
`FLVPlayback.CONNECTION_ERROR`	`"connectionError"`	Valeur permettant de tester la propriété d'état.
`FLVPlayback.DISCONNECTED`	`"disconnected"`	Valeur permettant de tester la propriété d'état.
`FLVPlayback.EVENT`	`"event"`	Peut être utilisée comme type de paramètre pour les méthodes `findCuePoint()` et `findNearestCuePoint()`.
`FLVPlayback.FLV`	`"flv"`	Peut être utilisée comme type de paramètre pour les méthodes `findCuePoint()` et `findNearestCuePoint()`.
`FLVPlayback.LOADING`	`"loading"`	Valeur permettant de tester la propriété d'état.
`FLVPlayback.NAVIGATION`	`"navigation"`	Peut être utilisée comme type de paramètre pour les méthodes `findCuePoint()` et `findNearestCuePoint()`.
`FLVPlayback.PAUSED`	`"paused"`	Valeur permettant de tester la propriété d'état.

Propriété	Valeur	Description
FLVPlayback.PLAYING	"playing"	Valeur permettant de tester la propriété d'état.
FLVPlayback.REWINDING	"rewinding"	Valeur permettant de tester la propriété d'état.
FLVPlayback.SEEKING	"seeking"	Valeur permettant de tester la propriété d'état.
FLVPlayback.STOPPED	"stopped"	Valeur permettant de tester la propriété d'état.
FLVPlayback.version	Nombre correspondant au numéro de version du composant	Pour déterminer la version du composant FLVPlayback que vous utilisez.

Propriétés d'occurrence

Le tableau ci-dessous répertorie les propriétés d'occurrence de la classe FLVPlayback. Cet ensemble de propriétés s'applique à chaque occurrence d'un composant FLVPlayback de l'application. Par exemple, si vous faites glisser deux occurrences du composant FLVPlayback sur la scène, chacune possède un ensemble de ces propriétés.

Propriété	Description
FLVPlayback.activeVideoPlayerIndex	Nombre qui indique le flux continu du fichier FLV affecté par d'autres méthodes, propriétés et événements. Utilisez cette propriété pour gérer plusieurs flux continus de fichier FLV ; la valeur par défaut est 0.
FLVPlayback.autoPlay	Valeur booléenne qui, si elle est définie sur `true`, précise que le composant lit le fichier FLV dès son chargement. La valeur par défaut est `true`.
FLVPlayback.autoRewind	Valeur booléenne qui, si elle est définie sur `true` provoque le rembobinage du fichier FLV jusqu'à l'image 1 à l'arrêt de la lecture.
FLVPlayback.autoSize	Valeur booléenne qui, si elle est définie sur `true`, redimensionne automatiquement la vidéo aux dimensions sources.

Propriété	Description
FLVPlayback.backButton	Objet MovieClip correspondant à la commande BackButton.
FLVPlayback.bitrate	Nombre qui indique la bande passante de l'utilisateur en bits par seconde. Utilisée dans certains cas pour choisir le fichier FLV à lire.
FLVPlayback.buffering	Valeur booléenne définie sur `true` si la vidéo est dans un état de mise en mémoire tampon. Lecture seule.
FLVPlayback.bufferingBar	Objet MovieClip correspondant à la commande BufferingBar.
FLVPlayback.bufferingBarHidesAndDisablesOthers	Affecte le comportement des commandes lorsque le composant entre dans l'état de mise en mémoire tampon.
FLVPlayback.bufferTime	Nombre indiquant le nombre de secondes à mettre en mémoire tampon avant de commencer la lecture d'un flux vidéo.
FLVPlayback.bytesLoaded	Nombre indiquant le degré de téléchargement en nombre d'octets pour un téléchargement HTTP. Lecture seule.
FLVPlayback.bytesTotal	Nombre total d'octets téléchargés pour un téléchargement HTTP. Lecture seule.
FLVPlayback.contentPath	Chaîne indiquant l'URL d'un fichier FLV à charger.
FLVPlayback.cuePoints	Tableau décrivant des points de repère ActionScript et des points de repère intégrés au fichier FLV désactivés. Ne doit jamais être utilisée directement dans ActionScript. A définir uniquement dans la boîte de dialogue Points de repère.
FLVPlayback.forwardButton	Objet MovieClip correspondant à la commande ForwardButton.
FLVPlayback.height	Nombre indiquant la hauteur de la vidéo en pixels.

Propriété	Description
`FLVPlayback.idleTimeout`	Temps écoulé (en millisecondes) avant que Flash mette fin à une connexion inactive au serveur FMS en raison de l'interruption ou de l'arrêt de la lecture.
`FLVPlayback.isLive`	Valeur booléenne définie sur `true` si le flux vidéo est en direct.
`FLVPlayback.isRTMP`	Valeur booléenne définie sur `true` si le fichier FLV est diffusé en continu à partir d'un serveur FMS ou d'un serveur FVSS. Lecture seule.
`FLVPlayback.maintainAspectRatio`	Valeur booléenne qui, si elle est définie sur `true`, conserve les proportions de la vidéo.
`FLVPlayback.metadata`	Objet représentant un paquet d'informations de métadonnées reçu à partir d'un appel à la fonction de rappel `onMetaData()`, le cas échéant. Lecture seule.
`FLVPlayback.metadataLoaded`	Valeur booléenne définie sur `true` si un paquet de métadonnées a été trouvé et traité ou s'il est certain qu'il ne le sera pas. Lecture seule.
`FLVPlayback.muteButton`	Objet MovieClip correspondant à la commande MuteButton.
`FLVPlayback.ncMgr`	Objet INCManager fournissant l'accès à une occurrence de la classe implémentant INCManager Lecture seule.
`FLVPlayback.pauseButton`	Objet MovieClip correspondant à la commande PauseButton.
`FLVPlayback.paused`	Valeur booléenne définie sur `true` si le fichier FLV est dans un état de pause. Lecture seule.
`FLVPlayback.playButton`	Objet MovieClip correspondant à la commande PlayButton.
`FLVPlayback.playheadPercentage`	Nombre indiquant la durée de lecture comme pourcentage de la durée totale du fichier FLV.

Propriété	Description
FLVPlayback.playheadTime	Nombre représentant la durée de lecture (ou position de la tête de lecture) actuelle, mesurée en secondes, qui peut être une valeur décimale.
FLVPlayback.playheadUpdateInterval	Nombre représentant l'intervalle (en millisecondes) entre chaque événement `playheadUpdate`.
FLVPlayback.playing	Valeur booléenne définie sur `true` si le fichier FLV est en cours de lecture. Lecture seule.
FLVPlayback.playPauseButton	Objet MovieClip correspondant à la commande PlayPauseButton.
FLVPlayback.preferredHeight	Nombre indiquant la hauteur du fichier FLV source.
FLVPlayback.preferredWidth	Nombre indiquant la largeur du fichier FLV source.
FLVPlayback.progressInterval	Nombre représentant l'intervalle (en millisecondes) entre chaque événement progress.
FLVPlayback.scaleX	Nombre indiquant le redimensionnement horizontal.
FLVPlayback.scaleY	Nombre indiquant le redimensionnement vertical.
FLVPlayback.scrubbing	Valeur booléenne définie sur `true` si l'utilisateur fait glisser la poignée seekBar. Lecture seule.
FLVPlayback.seekBar	Objet MovieClip correspondant à la commande SeekBar.
FLVPlayback.seekBarInterval	Nombre indiquant la fréquence (en millisecondes) à laquelle la poignée seekBar est vérifiée lors de la modulation. La valeur par défaut est 250.
FLVPlayback.seekBarScrubTolerance	Nombre indiquant (en pourcentage) jusqu'où l'utilisateur peut déplacer la poignée scrubBar avant une mise à jour. La valeur est indiqué en pourcentage, de 1 à 100.

Propriété	Description
FLVPlayback.seekToPrevOffset	Nombre (en secondes) utilisé par la méthode seekToPrevNavCuePoint() pour comparer son heure avec le point de repère précédent.
FLVPlayback.skin	Chaîne indiquant le nom d'un fichier SWF d'enveloppe.
FLVPlayback.skinAutoHide	Valeur booléenne qui, si elle est définie sur true, masque l'enveloppe du composant lorsque la souris n'est pas placée sur la vidéo. La valeur par défaut est false.
FLVPlayback.state	Chaîne indiquant l'état du composant. Définie avec les méthodes load(), play(), stop(), pause() et seek(). Lecture seule.
FLVPlayback.stateResponsive	Valeur booléenne définie sur true si l'état est réactif. Lecture seule.
FLVPlayback.stopButton	Objet MovieClip correspondant à la commande StopButton.
FLVPlayback.stopped	Valeur booléenne définie sur true si l'état est arrêté. Lecture seule.
FLVPlayback.totalTime	Nombre représentant la durée de lecture totale de la vidéo en secondes.
FLVPlayback.transform	Objet permettant d'accéder directement aux méthodes Sound.setTransform() et Sound.getTransform() afin de fournir un meilleur son.
FLVPlayback.visible	Valeur booléenne qui, si elle est définie sur true, affiche le composant FLVPlayback.
FLVPlayback.visibleVideoPlayerIndex	Nombre que vous pouvez utiliser pour gérer plusieurs flux de fichier FLV. Indique l'occurrence de lecteur vidéo visible et audible. La valeur par défaut est 0.
FLVPlayback.volume	Nombre compris entre 0 et 100 indiquant le réglage de contrôle du volume.

Propriété	Description
FLVPlayback.volumeBar	Objet MovieClip correspondant à la commande VolumeBar.
FLVPlayback.volumeBarInterval	Nombre indiquant la fréquence (en millisecondes) à laquelle la poignée volumeBar est vérifiée lors de la modulation. La valeur par défaut est 250.
FLVPlayback.volumeBarScrubTolerance	Nombre indiquant (en pourcentage) jusqu'où l'utilisateur peut déplacer la poignée de la barre de volume avant qu'une mise à jour se produise.
FLVPlayback.width	Nombre indiquant la largeur de l'occurrence de composant en pixels.
FLVPlayback.x	Nombre indiquant l'emplacement à l'horizontale du lecteur vidéo en pixels.
FLVPlayback.y	Nombre indiquant l'emplacement à la verticale du lecteur vidéo en pixels.

Récapitulatif des événements de la classe FLVPlayback

Le tableau suivant répertorie les événements de la classe FLVPlayback.

Evénement	Description
FLVPlayback.buffering	Distribué lorsque le composant entre dans l'état de mise en mémoire tampon.
FLVPlayback.close	Distribué lorsque `NetConnection` est fermé, via un timeout ou un appel à la méthode `close()`.
FLVPlayback.complete	Distribué lorsque la lecture se termine en atteignant la fin du fichier FLV.
FLVPlayback.cuePoint	Distribué lorsqu'un point de repère est atteint.
FLVPlayback.fastForward	Distribué lorsque l'emplacement de la tête de lecture est avancé suite à un appel à la méthode `seek()`.
FLVPlayback.metadata	Distribué la première fois que les métadonnées du fichier FLV sont atteintes.
FLVPlayback.paused	Distribué lorsque le composant entre dans l'état de pause.

Evénement	Description
FLVPlayback.playheadUpdate	Distribué toutes les 0,25 secondes lors de la lecture du fichier FLV. La propriété playheadUpdateInterval permet de spécifier la fréquence.
FLVPlayback.playing	Distribué lorsque le composant entre dans l'état de lecture.
FLVPlayback.progress	Distribué toutes les 0,25 secondes, en commençant lorsque la méthode load() est appelée et en se terminant lorsque tous les octets sont chargés ou qu'il existe une erreur réseau. La propriété progressInterval permet de spécifier la fréquence.
FLVPlayback.ready	Distribué lorsque le fichier FLV est chargé et prêt à être affiché.
FLVPlayback.resize	Distribué lors du redimensionnement de la vidéo.
FLVPlayback.rewind	Distribué lorsque l'emplacement de la tête de lecture recule suite à un appel à la méthode seek() ou à la fin du rembobinage automatique.
FLVPlayback.scrubFinish	Distribué lorsque l'utilisateur arrête la modulation du scénario avec SeekBar.
FLVPlayback.scrubStart	Distribué lorsque l'utilisateur commence la modulation du scénario avec SeekBar.
FLVPlayback.seek	Distribué lorsque l'emplacement de la tête de lecture est modifié suite à un appel à la méthode seek() ou en utilisant la commande correspondante.
FLVPlayback.skinError	Distribué lorsqu'une erreur se produit lors du chargement d'un fichier SWF d'enveloppe.
FLVPlayback.skinLoaded	Distribué lors du chargement d'un fichier SWF d'enveloppe.
FLVPlayback.stateChange	Distribué lorsque l'état de lecture est modifié.
FLVPlayback.stopped	Distribué lorsque le composant entre dans l'état arrêté.
FLVPlayback.volumeUpdate	Distribué lorsque le volume est modifié au moyen de la propriété volume

FLVPlayback.ACTIONSCRIPT

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`mx.video.FLVPlayback.ACTIONSCRIPT`

Description

Propriété de la classe FLVPlayback en lecture seule qui contient la constante de type chaîne, « `actionscript` », à utiliser comme propriété `type` avec les méthodes `findCuePoint()` et `findNearestCuePoint()`.

Exemple

L'exemple ci-dessous utilise la constante `ACTIONSCRIPT` afin de définir la propriété `type` de la méthode `findCuePoint()`.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object) {
  var cuePt:Object = new Object(); // Création d'un objet point de repère.
  cuePt.time = 2.444;
  cuePt.name = "ASCuePt1";
  my_FLVPlybk.addASCuePoint(cuePt);  // Ajout d'un point de repère AS.
}
my_FLVPlybk.addEventListener("ready", listenerObject)
listenerObject.playing = function(eventObject:Object) {
    var rtn_obj:Object = new Object();
    if(rtn_obj = my_FLVPlybk.findCuePoint(2.444,
  FLVPlayback.ACTIONSCRIPT)){
      trace("Found cue point " + rtn_obj.name);
    }
  }
my_FLVPlybk.addEventListener("playing", listenerObject)
```

Voir aussi

`FLVPlayback.findCuePoint()`

FLVPlayback.activeVideoPlayerIndex

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.activeVideoPlayerIndex

Description

Propriété : nombre qui indique l'occurrence de lecteur vidéo affectée par les autres API. Utilisez cette propriété pour gérer plusieurs flux continus de fichier FLV. La valeur par défaut est 0.

Cette propriété n'affiche pas le lecteur vidéo. Pour ce faire, utilisez la propriété `visibleVideoPlayerIndex`.

Un lecteur vidéo est créé la première fois que la propriété `activeVideoPlayer` est définie sur un nombre. Lorsque le lecteur vidéo est créé, ses propriétés sont définies sur la valeur du lecteur vidéo par défaut (`activeVideoPlayerIndex == 0`) excepté les propriétés `contentPath`, `totalTime` et `isLive`, qui sont toujours définies sur les valeurs par défaut (chaîne vide, 0 et `false`, respectivement) et la propriété `autoPlay` qui est toujours définie sur `false` (la valeur par défaut est `true` uniquement pour le lecteur vidéo par défaut, `0`).
La propriété `cuePoints` n'a aucun effet, et n'a pas d'incidences sur un chargement ultérieur dans le lecteur vidéo par défaut.

Les API qui contrôlent le volume, le positionnement, les dimensions, la visibilité et les commandes d'interface utilisateur sont toujours globales et leur comportement n'est *pas* affecté par la définition de la propriété `activeVideoPlayerIndex`. De façon spécifique, la définition de cette propriété n'a aucun effet sur les propriétés et les méthodes répertoriées ci-dessous.

Propriétés et méthodes non affectées par la propriété activeVideoPlayerIndex

`backButton`	`playPauseButton`	`skin`	`width`
`bufferingBar`	`scaleX`	`stopButton`	`x`
`bufferingBarHidesAndDisablesOthers`		`transform`	`y`
`forwardButton`	`scaleY`	`visible`	`setSize()`
`height`	`seekBar`	`volume`	`setScale()`
`muteButton`	`seekBarInterval`	`volumeBar`	
`pauseButton`	`seekBarScrubTolerance`	`volumeBarInterval`	
`playButton`	`seekToPrevOffset`	`volumeBarScrubTolerance`	

REMARQUE

La propriété `visibleVideoPlayerIndex` et *non* la propriété `activeVideoPlayerIndex` détermine le lecteur vidéo contrôlé par l'enveloppe.

Cependant, les API qui contrôlent les dimensions interagissent avec la propriété `visibleVideoPlayerIndex`. Pour plus d'informations, voir « FLVPlayback.visibleVideoPlayerIndex », à la page 713.

Les API restantes ciblent un lecteur vidéo spécifique en fonction de la définition de la propriété `activeVideoPlayerIndex`.

Lorsque vous écoutez des événements, vous obtenez tous les événements correspondant à l'ensemble des lecteurs vidéo. Pour savoir à quel lecteur vidéo correspond l'événement, utilisez la propriété `vp` de l'événement. Il s'agit d'un nombre correspondant à celui défini dans `activeVideoPlayerIndex` et `visibleVideoPlayerIndex`. Tous les événements ont cette propriété *excepté* `resize` et `volume`, qui ne sont pas propres à un lecteur vidéo, mais sont globaux pour l'occurrence du composant FLVPlayback.

Par exemple, pour charger un deuxième fichier FLV dans l'arrière-plan, définissez `activeVideoPlayerIndex` sur 1 et appelez la méthode `load()`. Lorsque vous souhaitez afficher celui-ci et masquer le premier, définissez `visibleVideoPlayerIndex` sur 1.

Exemple

L'exemple suivant crée deux lecteurs vidéo pour lire deux fichiers FLV de manière consécutive dans une même occurrence de fichier FLV. Il définit la propriété `activeVideoPlayerIndex` afin de basculer entre les lecteurs vidéo et leurs fichiers FLV respectifs.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
// Indication du nom et de l'emplacement du fichier FLV pour le lecteur
// par défaut.
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  clouds.flv";
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  // Ajout d'un second lecteur vidéo et indication du nom et
  // de l'emplacement de son fichier FLV.
  my_FLVPlybk.activeVideoPlayerIndex = 1;
  my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
  // Réinitialisation sur le lecteur vidéo par défaut qui lit
  // automatiquement son fichier FLV.
  my_FLVPlybk.activeVideoPlayerIndex = 0;
};
my_FLVPlybk.addEventListener("ready", listenerObject);
listenerObject.complete = function(eventObject:Object):Void {
  // Si l'événement complete concerne le second fichier FLV,
  // rendez-le actif et visible par défaut.
  if (eventObject.vp == 1) {
    my_FLVPlybk.activeVideoPlayerIndex = 0;
    my_FLVPlybk.visibleVideoPlayerIndex = 0;
  }
  else { // Activation et affichage du second lecteur, lecture
         // du fichier FLV.
    my_FLVPlybk.activeVideoPlayerIndex = 1;
    my_FLVPlybk.visibleVideoPlayerIndex = 1;
    my_FLVPlybk.play();
  }
};
// Ajout d'un écouteur pour un événement complete.
my_FLVPlybk.addEventListener("complete", listenerObject);
```

Voir aussi

`FLVPlayback.bringVideoPlayerToFront()`, `FLVPlayback.getVideoPlayer()`, `Classe VideoPlayer`, `FLVPlayback.visibleVideoPlayerIndex`

FLVPlayback.addASCuePoint()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
my_FLVplybk.addASCuePoint(cuePoint:Object)
my_FLVplybk.addASCuePoint(time:Number, name:String[, parameters:Object])
```

Paramètres

CuePoint Objet possédant les propriétés *name*:String et *time*:Number (en secondes), qui décrivent le point de repère. Il peut également être doté d'une propriété *parameters*:Object qui contient des paires nom/valeur. Il peut avoir une propriété *type*:String définie sur « actionscript ». Si le paramètre *type* est absent ou défini autrement, cette propriété est définie automatiquement. Si l'objet n'est pas conforme à ces conventions, la méthode renvoie une erreur VideoError.

time Nombre indiquant l'heure du nouveau point de repère à ajouter. Si vous utilisez le paramètre *time*, le paramètre *name* doit suivre.

name Chaîne indiquant le nom du point de repère si vous soumettez un paramètre *time* au lieu de l'objet *CuePoint*.

parameters Paramètres facultatifs du point de repère.

Renvoie

Copie de l'objet point de repère ajouté avec les propriétés supplémentaires suivantes :

- array Tableau des points de repère cherchés. Considérez ce tableau comme étant en lecture seule, car si vous y ajoutez, supprimez ou modifiez des objets, les points de repère risquent de ne pas fonctionner correctement.
- index Index dans le tableau pour le point de repère renvoyé.

Description

Méthode : ajoute un point de repère ActionScript et a le même effet que l'ajout d'un point de repère ActionScript à l'aide de la boîte de dialogue Points de repère, excepté que cette méthode se produit lors de l'exécution d'une application plutôt que lors de son développement.

Les informations de point de repère sont effacées lorsque la propriété contentPath est définie. Pour définir des informations de point de repère relatives au fichier FLV suivant à charger, définissez d'abord la propriété contentPath.

Il est possible d'ajouter plusieurs points de repère ActionScript avec le même nom et la même heure. Si vous supprimez des points de repère ActionScript à l'aide de la méthode `removeASCuePoint()`, tous les points de repère avec le même nom et la même heure sont supprimés.

Exemple

L'exemple suivant ajoute deux points de repère ActionScript à un fichier FLV. Le premier est ajouté à l'aide du paramètre *`cuePoint`* et le second en utilisant les paramètres *`time`* et *`name`*. Lorsque chaque point de repère se présente pendant la lecture, un écouteur d'événements cuePoint affiche la valeur de la propriété `playheadTime` dans une zone de texte.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Faites glisser un composant TextArea sur la scène, sous l'occurrence FLVPlayback, et nommez l'occurrence, TextArea **my_ta**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
  - Composant TextArea sur la scène dont l'occurrence est nommée my_ta
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
my_ta.visible = false;
// Création d'un objet point de repère.
var cuePt:Object = new Object(); // Création d'un objet point de repère.
cuePt.time = 2.444;
cuePt.name = "ASCuePt1";
cuePt.type = "actionscript";
my_FLVPlybk.addASCuePoint(cuePt);  // Ajout d'un point de repère AS.
// Ajout du second point de repère AS à l'aide des paramètres d'heure et
// de nom.
my_FLVPlybk.addASCuePoint(5, "ASCuePt2");
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object):Void {
    my_ta.text = "Elapsed time in seconds: " + my_FLVPlybk.playheadTime;
    my_ta.visible = true;
};
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
```

Voir aussi

`FLVPlayback.findCuePoint()`, `FLVPlayback.removeASCuePoint()`

FLVPlayback.addEventListener()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation:

```
my_FLVPlybk.addEventListener(event:String, listener:Object):Void
my_FLVPlybk.addEventListener(event:String, listener:Function):Void
```

Paramètres

event Chaîne qui indique le nom de l'événement pour lequel vous enregistrez un écouteur. Si cet écouteur est un objet, il s'agit également du nom de la fonction d'objet écouteur à appeler.

listener Nom de l'object écouteur ou de la fonction d'écouteur que vous enregistrez pour cet événement.

Renvoie

Aucune.

Description

Méthode : enregistre un objet écouteur ou une fonction d'écouteur pour un événement spécifié. Si cet écouteur est un objet, il doit posséder une fonction définie pour lui dont le nom est identique à celui de l'événement. Si cet écouteur est une fonction, c'est son nom qui sera appelé pour gérer l'événement.

Exemple

L'exemple suivant écoute un événement `complete` et affiche un message dans une zone de texte, le cas échéant.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Faites glisser un composant TextArea sur la scène, sous l'occurrence FLVPlayback, et nommez l'occurrence, TextArea **my_ta**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
Usage 1:: listener object
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
  - Composant TextArea sur la scène dont l'occurrence est nommée my_ta
*/
```

```
import mx.video.*;
my_ta.visible = false;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object(); // Création d'un objet écouteur.
listenerObject.complete = function(eventObject:Object):Void {
    my_ta.text = "That's All Folks!";
    my_ta.visible = true;
};
my_FLVPlybk.addEventListener("complete", listenerObject);
```

Usage 2: listener function

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
  - Composant TextArea sur la scène dont l'occurrence est nommée my_ta
*/
import mx.video.*;
my_ta.visible = false;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
function the_end(eventObject:Object):Void {
    my_ta.text = "That's All Folks!";
    my_ta.visible = true;
};
my_FLVPlybk.addEventListener("complete", the_end);
```

Voir aussi

Récapitulatif des événements de la classe FLVPlayback,
FLVPlayback.removeEventListener(),

FLVPlayback.ALL

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
mx.video.FLVPlayback.ALL
```

Description

Propriété de la classe FLVPlayback en lecture seule qui contient la constante de type chaîne « `all` ». Vous pouvez utiliser cette propriété comme paramètre `type` des méthodes `findCuePoint()` et `findNearestCuePoint()`.

Exemple

L'exemple suivant recherche un point de repère nommé `point2` dont l'heure est définie sur 7.748 parmi tous les points de repère. Il affiche les propriétés `type` et `time` du point de repère trouvé.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
// Création d'un objet point de repère.
var listenerObject = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  var cuePt:Object = new Object(); // Création d'un objet point de repère.
  cuePt.name = "point2";
  cuePt.time = 7.748;
  if(cuePt = my_FLVPlybk.findCuePoint(cuePt, FLVPlayback.ALL)) // Recherche
                                                        // d'un point de repère.
    trace("found a " + cuePt.type + " cue point at " + cuePt.time);
  else
    trace("cue point not found");
}
my_FLVPlybk.addEventListener("ready", listenerObject);
```

Voir aussi

`FLVPlayback.findCuePoint()`

FLVPlayback.autoPlay

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.autoPlay

Description

Propriété : valeur booléenne qui, si elle est définie sur `true`, provoque la lecture immédiate du fichier FLV lorsque la propriété `contentPath` est définie. Si elle est définie sur `false`, le composant attend la commande de lecture. Même si `autoPlay` est défini sur `false`, le composant charge le contenu immédiatement. La valeur par défaut est `true`.

Si vous définissez la propriété sur `true` entre le chargement de nouveaux fichiers FLV, il n'y a aucune incidence tant que la propriété `contentPath` n'est pas définie.

Exemple

L'exemple suivant désactive la lecture du fichier FLV pour définir 30 % de la tête de lecture dans l'heure de lecture et commencer la lecture à ce point.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.autoPlay = false;
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  my_FLVPlybk.seekPercent(30);
  my_FLVPlybk.play();
};
my_FLVPlybk.addEventListener("ready", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.activeVideoPlayerIndex`

FLVPlayback.autoRewind

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.autoRewind

Description

Propriété : valeur booléenne qui, si elle est définie sur `true`, provoque le rembobinage du fichier FLV jusqu'à l'image 1 lorsque la lecture s'arrête, soit parce que le lecteur a atteint la fin du flux, soit parce que la méthode `stop()` a été appelée. Cette propriété n'a aucune signification pour les flux en direct. La valeur par défaut est `true`.

Exemple

L'exemple suivant définit la propriété `autoRewind` sur `false` pour empêcher le rembobinage automatique du fichier FLV à la fin de la lecture.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback dans la bibliothèque
*/
my_FLVPlybk.autoRewind = false;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

FLVPlayback.autoSize

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.autoSize

Description

Propriété : valeur booléenne qui, si elle est définie sur `true`, dimensionne automatiquement la vidéo aux dimensions du fichier FLV source. Si vous définissez cette propriété de `false` à `true` après le chargement d'un fichier FLV, le redimensionnement automatique démarre immédiatement. La valeur par défaut est `false`.

Exemple

L'exemple suivant affiche d'abord les dimensions source du fichier FLV (`preferredWidth` et `preferredHeight`) lorsqu'il est prêt pour la lecture. Il appelle ensuite la méthode `setSize()` pour changer les dimensions de l'occurrence FLVPlayback, déclenchant un événement `resize`. Il définit la propriété `autoSize` sur `true`, déclenchant alors un autre événement `resize` qui rétablit les dimensions du fichier FLV source. Le gestionnaire d'événements `resize` affiche les dimensions de l'occurrence FLVPlayback dans le panneau Sortie après chaque événement.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback dans la bibliothèque
*/
import mx.video.*;
my_FLVPlybk.maintainAspectRatio = false;
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object) {
  trace("FLV width is: " + my_FLVPlybk.preferredWidth + " FLV height is: "
  + my_FLVPlybk.preferredHeight);
  my_FLVPlybk.setSize(400, 400);
  my_FLVPlybk.autoSize = true;
};
my_FLVPlybk.addEventListener("ready", listenerObject);
listenerObject.resize = function(eventObject:Object) {
  trace("my_FLVPlybk width is: " + my_FLVPlybk.width + ";
  my_FLVPlybk.height is: " + my_FLVPlybk.height);
};
my_FLVPlybk.addEventListener("resize", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.maintainAspectRatio`, `FLVPlayback.preferredHeight`, `FLVPlayback.preferredWidth`, `FLVPlayback.resize`

FLVPlayback.backButton

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.backButton

Description

Propriété : objet MovieClip qui correspond à la commande de lecture BackButton. Pour plus d'informations sur l'utilisation des composants de l'interface utilisateur personnalisée Lecture de fichiers FLV pour les commandes de lecture, reportez-vous à « Application d'enveloppe aux composants particuliers de l'interface utilisateur personnalisée Lecture de fichiers FLV », à la page 544.

Cliquer sur la commande BackButton provoque un événement `rewind`.

Exemple

L'exemple suivant utilise les propriétés `backButton`, `forwardButton`, `playButton`, `pauseButton` et `stopButton` pour associer des commandes individuelles de l'interface utilisateur personnalisée Lecture de fichiers FLV à un composant FLVPlayback.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Dans l'inspecteur des composants, définissez le paramètre Skin (Enveloppe) sur None. Ajoutez ensuite les composants individuels suivants de l'interface utilisateur personnalisée Lecture de fichiers FLV et donnez-leur les noms d'occurrence indiqués entre parenthèses : BackButton (**my_bkbttn**), ForwardButton (**my_fwdbttn**), PlayButton (**my_plybttn**), PauseButton (**my_pausbttn**) et StopButton (**my_stopbttn**). Ajoutez ensuite les lignes de code suivantes au panneau Actions :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
  - Composants BackButton, ForwardButton, PlayButton, PauseButton et
   StopButton de l'interface utilisateur personnalisée FLV dans la
  bibliothèque
*/
import mx.video.*;
my_FLVPlybk.backButton = my_bkbttn;
my_FLVPlybk.forwardButton = my_fwdbttn;
my_FLVPlybk.playButton = my_plybttn;
```

```
my_FLVPlybk.pauseButton = my_pausbttn;
my_FLVPlybk.stopButton = my_stopbttn;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.forwardButton`, `FLVPlayback.rewind`

FLVPlayback.bitrate

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVplybk.bitrate

Description

Propriété : nombre qui spécifie la vitesse de transmission (bits par seconde) à laquelle transférer le fichier FLV.

Lors d'un téléchargement progressif, vous pouvez utiliser le format SMIL, mais vous devez définir la vitesse de transmission, car il n'existe pas de détection automatique.

Pour diffuser la vidéo en continu à partir d'un serveur FMS, vous pouvez indiquer un fichier SMIL qui décrit comment basculer entre plusieurs flux continus en fonction de la bande passante. La bande passante est détectée automatiquement par FMS et dans ce cas, `bitrate` est ignoré (s'il est défini).

Pour plus d'informations sur l'utilisation d'un fichier SMIL, reportez-vous à « Utilisation d'un fichier SMIL », à la page 737.

Exemple

L'exemple suivant active deux boutons radio pour déterminer la vitesse de transmission à utiliser lorsque vous sélectionnez un fichier FLV dans le fichier SMIL spécifié. Les balises `video` adéquates du fichier SMIL sont indiquées dans le code suivant :

```
<switch>
  <video src="myvideo_mdm.flv" system-bitrate="56000"  dur="3:00.1">
  <video src="myvideo_isdn.flv" dur="3:00.1">
</switch>
```

S'il s'agit d'une connexion lente, le code définit la propriété `bitrate` pour forcer la sélection du fichier FLV approprié. S'il s'agit de connexions plus rapides, il tire parti de la détection automatique de la bande passante pour la diffusion en continu à partir de FMS, et ne définit pas la vitesse de transmission.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Faites glisser un composant RadioButton, puis un composant Label vers le panneau Bibliothèque. Ajoutez ensuite le code suivant dans le panneau Actions, sur l'image 1 du scénario. Dans l'instruction qui charge la propriété `contentPath`, remplacez le texte en italique par le nom et l'emplacement de votre fichier SMIL.

```
/**
 Requiert :
  - Composant FLVPlayback dans la bibliothèque
  - Composant RadioButton dans la bibliothèque
  - Composant Label dans la bibliothèque
*/
import mx.video.*;
import mx.controls.*;

this.createClassObject(Label, "my_prompt", 10);
my_prompt.text = "Please indicate your connection speed: ";
this.createClassObject(RadioButton, "dialup", 20, {label:"Dialup modem
  (56 KB)", groupName:"radioGroup"});
this.createClassObject(RadioButton, "isdn", 30, {label:"ISDN (128 KB)",
  groupName:"radioGroup"});
my_prompt.autoSize = "left";
dialup.setSize(200, 30);
isdn.setSize(200, 30);
// Positionnement des boutons radio sur la scène.
my_prompt.move(my_FLVPlybk.x, my_FLVPlybk.y + my_FLVPlybk.height + 40);
dialup.move(my_FLVPlybk.x, my_prompt.y + my_prompt.height + 5);
isdn.move(my_FLVPlybk.x, dialup.y + 15);

// Création d'un objet écouteur
var rbListener:Object = new Object();
rbListener.click = function(eventObject:Object){
   if(dialup.selected) {   // pour modem.
     my_FLVPlybk.bitrate = 56000;
   } // pour RNIS (ou bandes passantes plus élevées) - détection
     // automatique autorisée.
}
// Ajout de l'écouteur.
radioGroup.addEventListener("click", rbListener);
my_FLVPlybk.contentPath = "http://www.someserver.com/video/sample.smil";
```

FLVPlayback.bringVideoPlayerToFront()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.bringVideoPlayerToFront(*index*:Number)

Paramètres

index Nombre correspondant à l'index du lecteur vidéo à déplacer vers l'avant.

Description

Méthode : place un lecteur vidéo devant la pile des lecteurs vidéo. Utile pour les transitions personnalisées entre lecteurs vidéo. L'ordre de la pile est identique à celui de la propriété `activeVideoPlayerIndex` : 0 est en bas, 1 est au-dessus de 0, 2 est au-dessus de 1, etc.

Exemple

L'exemple suivant utilise deux lecteurs vidéo pour lire deux fichiers FLV. Lorsque chacun des trois points de repère du premier fichier FLV (cuepoints.flv) est trouvé, l'exemple appelle la méthode `bringVideoPlayerToFront()` pour placer l'autre fichier FLV à l'avant. Comme l'exemple définit la propriété `_alpha` sur 75 pour le lecteur vidéo numéro 1, le fichier FLV (plane_cuepoints) lu sur ce lecteur est transparent, ce qui rend simultanément les deux fichiers FLV visibles lorsque ce fichier FLV est devant.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
my_FLVPlybk.load("http://www.helpexamples.com/flash/video/cuepoints.flv");
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object) {
    if (eventObject.target.contentPath == "http://www.helpexamples.com/
  flash/video/cuepoints.flv") {
        // Déclenchement une fois que le premier fichier FLV est prêt.
        my_FLVPlybk.activeVideoPlayerIndex = 1;
        my_FLVPlybk.load("http://www.helpexamples.com/flash/video/
  plane_cuepoints.flv");
    } else {
```

```
        // Déclenchement une fois que le second fichier FLV est prêt.
        eventObject.target.activeVideoPlayerIndex = 0;
        eventObject.target.play();
        eventObject.target.activeVideoPlayerIndex = 1;
        eventObject.target.play();
        var layerOnTop:MovieClip = eventObject.target.getVideoPlayer(1);
        layerOnTop._alpha = 75;
        layerOnTop._visible = true;
    }
}
my_FLVPlybk.addEventListener("ready", listenerObject);

var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object) {

    // En fonction du nom du point, placer l'un ou l'autre devant.
    if (eventObject.info.name == "point1") {
     trace(eventObject.info.name + " : 0 to front");
        eventObject.target.bringVideoPlayerToFront(1);
    } else if (eventObject.info.name == "point2") {
     trace(eventObject.info.name + " : 1 to front");
        eventObject.target.bringVideoPlayerToFront(0);
    }else if (eventObject.info.name == "point3") {
     trace(eventObject.info.name + " : 0 to front");
        eventObject.target.bringVideoPlayerToFront(1);
    }
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
```

Voir aussi

FLVPlayback.activeVideoPlayerIndex, FLVPlayback.getVideoPlayer(), Classe VideoPlayer, FLVPlayback.visibleVideoPlayerIndex

FLVPlayback.buffering

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.buffering = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("buffering", listenerObject);
```

Description

Evénement : distribué lorsque l'occurrence du composant FLVPlayback entre dans l'état de mise en mémoire tampon. L'occurrence FLVPlayback entre généralement dans cet état immédiatement après un appel à la méthode `play()` ou lorsque vous cliquez sur la commande correspondante, avant d'entrer dans l'état de lecture. L'objet événement possède les propriétés `state`, `playheadTime`. et `vp`, qui correspond au numéro d'index du lecteur vidéo auquel cet événement s'applique. Reportez-vous à « FLVPlayback.activeVideoPlayerIndex » à la page 571 et à « FLVPlayback.visibleVideoPlayerIndex » à la page 713.

L'occurrence FLVPlayback distribue également l'événement `stateChange` au début de la mise en mémoire tampon.

Exemple

L'exemple suivant crée un écouteur pour l'événement `buffering`. Lorsqu'un événement `buffering` se produit, le gestionnaire d'événements appelle la méthode `trace()` pour afficher les valeurs des propriétés `state` et `vp`.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.buffering = function(eventObject:Object) {
  trace("The state property has a value of " + eventObject.target.state);
  trace("The video player number is: " + eventObject.vp);
};
my_FLVPlybk.addEventListener("buffering", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.addEventListener()`, `FLVPlayback.state`, `FLVPlayback.removeEventListener()`

FLVPlayback.BUFFERING

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`mx.video.FLVPlayback.BUFFERING`

Description

Propriété : propriété de la classe FLVPlayback en lecture seule qui contient la constante de type chaîne « `buffering` ». Vous pouvez comparer cette propriété avec la propriété `state` pour déterminer si le composant est en état de mise en mémoire tampon.

Exemple

L'exemple suivant affiche la valeur de la propriété `FLVPlayback.BUFFERING` lorsque le composant entre dans un état de mise en mémoire tampon.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  if(eventObject.state == FLVPlayback.BUFFERING)
    trace(FLVPlayback.BUFFERING);
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.buffering

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.buffering

Description

Propriété : valeur booléenne définie sur `true` si la vidéo est dans un état de mise en mémoire tampon. Lecture seule.

Exemple

L'exemple suivant crée un écouteur pour l'événement `buffering` et affiche un message dans le panneau Sortie lorsque l'événement se produit.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  if(my_FLVPlybk.buffering){
    trace("The video is buffering");
  }
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.buffering`, `FLVPlayback.BUFFERING`, `FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.bufferingBar

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.bufferingBar

Description

Propriété : objet MovieClip qui correspond à la commande de barre de mise en mémoire tampon. Cette commande affiche quand le fichier FLV est en état de chargement ou de mise en mémoire tampon. Pour plus d'informations sur l'utilisation des composants de l'interface utilisateur personnalisée Lecture de fichiers FLV pour les commandes de lecture, reportez-vous à « Application d'enveloppe aux composants particuliers de l'interface utilisateur personnalisée Lecture de fichiers FLV », à la page 544.

Exemple

L'exemple suivant associe des commandes de l'interface utilisateur personnalisée Lecture de fichiers FLV à un composant FLVPlayback en définissant les propriétés suivantes : `playPauseButton`, `stopButton`, `backButton`, `forwardButton` et `bufferingBar`. La barre de mise en mémoire tampon apparaît uniquement lorsque le fichier FLV effectue une mise en mémoire tampon avant le début de la lecture.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Dans l'inspecteur des composants, définissez le paramètre Skin (Enveloppe) sur None. Ajoutez les composants individuels suivants de l'interface utilisateur personnalisée Lecture de fichiers FLV et donnez-leur les noms d'occurrence indiqués entre parenthèses : PlayPauseButton (**my_plypausbttn**), StopButton (**my_stopbttn**), BackButton (**my_bkbttn**), ForwardButton (**my_fwdbttn**) et BufferingBar (**my_buffrgbar**). Ajoutez ensuite les lignes de code suivantes dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
  - Composants PlayPauseButton, StopButton, BackButton, ForwardButton et
  BufferingButton de l'interface utilisateur personnalisée Lecture de
  fichiers FLV dans la bibliothèque
*/
import mx.video.*;
```

```
my_FLVPlybk.playPauseButton = my_plypausbttn;
my_FLVPlybk.stopButton = my_stopbttn;
my_FLVPlybk.backButton = my_bkbttn;
my_FLVPlybk.forwardButton = my_fwdbttn;
my_FLVPlybk.bufferingBar = my_buffrgbar;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.bufferingBarHidesAndDisablesOthers`

FLVPlayback.bufferingBarHidesAndDisablesOthers

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.bufferingBarHidesAndDisablesOthers

Description

Propriété : si elle est définie sur `true`, masque la commande SeekBar et désactive les commandes Play, Pause, PlayPause, BackButton et ForwardButton lorsque le fichier FLV est en état de mise en mémoire tampon. Cela peut être utile pour empêcher un utilisateur d'utiliser ces commandes pour essayer d'accélérer la lecture du fichier FLV lors de son téléchargement ou de sa diffusion en flux continu sur une connexion lente.

Exemple

L'exemple suivant suppose que vous lisez un fichier FLV en flux continu à partir d'un FMS ou de FVSS. Il définit la propriété `bufferingBarHidesAndDisablesOthers` pour désactiver les commandes Play, Pause, PlayPause, BackButton et ForwardButton et pour masquer la commande SeekBar lors de la mise en mémoire tampon du fichier FLV.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez ensuite le code suivant dans le panneau Actions, sur l'image 1 du scénario. Dans l'instruction qui charge la propriété `contentPath`, remplacez le texte en italique par le nom et l'emplacement d'un fichier FLV sur votre serveur FMS.

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.bufferTime = 15;
my_FLVPlybk.bufferingBarHidesAndDisablesOthers = true;
my_FLVPlybk.contentPath = "rtmp://host_name/somefolder/vid_name.flv";
```

Voir aussi

`FLVPlayback.bufferingBar`

FLVPlayback.bufferTime

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.bufferTime

Description

Propriété : nombre indiquant le nombre de secondes à mettre en mémoire tampon avant de commencer la lecture d'un flux vidéo. S'il s'agit de fichiers FLV diffusés en flux continu via RTMP, qui ne sont pas téléchargés, mais seulement mis en mémoire tampon, il peut être important d'augmenter ce paramètre par rapport à sa valeur par défaut 0,1. S'il s'agit d'un fichier FLV téléchargé progressivement sur HTTP, il y a peu d'avantages à augmenter cette valeur, même si cela peut améliorer la qualité de visualisation d'une vidéo de grande qualité sur un ordinateur plus ancien, moins performant.

REMARQUE	Cette propriété n'indique pas la proportion du fichier FLV à télécharger avant de démarrer sa lecture.

Exemple

L'exemple suivant définit la durée de mémoire tampon sur 8 secondes pour un fichier FLV diffusé en flux continu à partir d'un FMS.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez ensuite le code suivant dans le panneau Actions, sur l'image 1 du scénario. Dans l'instruction qui charge la propriété `contentPath`, remplacez le texte en italique par le nom et l'emplacement d'un fichier FLV sur votre serveur FMS.

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.bufferTime = 8;
my_FLVPlybk.contentPath = "rtmp://host_name/somefolder/vid_name.flv";
```

FLVPlayback.bytesLoaded

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
my_FLVPlybk.bytesLoaded
```

Description

Propriété : nombre indiquant le degré de téléchargement en nombre d'octets pour un téléchargement HTTP. Renvoie la valeur –1 en cas d'absence de flux, lorsque le flux provient d'un serveur FMS ou si les informations ne sont pas encore disponibles. La valeur renvoyée est utile uniquement pour un téléchargement HTTP. Lecture seule.

Exemple

L'exemple suivant affiche la valeur initiale de la propriété `bytesLoaded` ainsi que sa valeur lorsque l'événement `ready` se produit.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  trace("State is " + eventObject.state + "; ready to play");
  // Affichage du nombre d'octets chargés à ce stade.
  trace("Bytes loaded: " + my_FLVPlybk.bytesLoaded);
};
my_FLVPlybk.addEventListener("ready", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
trace("Bytes loaded: " + my_FLVPlybk.bytesLoaded); // -1 si le chargement
                                                   // n'a pas commencé.
```

FLVPlayback.bytesTotal

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.bytesTotal

Description

Propriété ; chiffre indiquant le nombre total d'octets téléchargés pour un téléchargement HTTP. Renvoie la valeur -1 en cas d'absence de flux, lorsque le flux provient d'un serveur FMS ou si les informations ne sont pas encore disponibles. La valeur renvoyée est utile uniquement pour un téléchargement HTTP. Lecture seule.

Exemple

L'exemple suivant utilise la propriété `bytesTotal` pour afficher le nombre d'octets chargés pendant un téléchargement HTTP. Lorsque l'événement `metadataReceived` se produit, le gestionnaire d'événements affiche cette valeur dans la zone de texte my_ta.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Faites glisser un composant TextArea sur la scène, sous l'occurrence FLVPlayback, et nommez l'occurrence, **my_ta**. Ajoutez ensuite le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
  - Composant TextArea sur la scène dont l'occurrence est nommée my_ta
*/
import mx.video.*;
my_ta.visible = false;
var listenerObject:Object = new Object();
listenerObject.metadataReceived = function(eventObject:Object):Void {
    my_ta.text = "Loading: " + my_FLVPlybk.bytesTotal + " bytes.";
    my_ta.visible = true;
};
my_FLVPlybk.addEventListener("metadataReceived", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

FLVPlayback.close

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.close = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("close", listenerObject);
```

Description

Evénement : l'occurrence FLVPlayback distribue cet événement à la fermeture de NetConnection, en faisant expirer le délai ou via un appel à la méthode `closeVideoPlayer()`, ou lorsque vous appelez les méthodes `load()` ou `play()` ou définissez `contentPath` et provoquez la fermeture de la connexion RTMP. L'occurrence FLVPlayback distribue cet événement uniquement lors d'une diffusion en flux continu à partir de FMS ou de FVSS. L'objet événement a les propriétés `state` et `playheadTime`.

L'événement est doté de la propriété `vp` qui correspond au numéro d'index du lecteur vidéo auquel cet événement s'applique.

Exemple

L'exemple suivant suppose que vous lisez un fichier FLV en flux continu à partir d'un FMS ou de FVSS. Lorsque le fichier FLV est complet, un écouteur d'événement `complete` définit la propriété `contentPath` sur l'emplacement d'un nouveau fichier FLV, ce qui déclenche un événement `close` sur la connexion RTMP du premier fichier FLV. L'écouteur d'événement `close` affiche le numéro d'index du lecteur vidéo pour lequel l'événement s'est produit.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez ensuite le code suivant dans le panneau Actions, sur l'image 1 du scénario. Dans l'instruction qui charge la propriété `contentPath`, remplacez le texte en italique par le nom et l'emplacement d'un fichier FLV sur votre serveur FMS.

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
// Ecoute de l'événement close sur la connexion RTMP ;
// affichage de l'index du lecteur vidéo.
listenerObject.close = function(eventObject:Object) {
  trace("Closed connection for video player: " + eventObject.vp);
};
my_FLVPlybk.addEventListener("close", listenerObject);
// Ecoute de l'événement complete ; lecture d'un nouveau fichier FLV.
listenerObject.complete = function(eventObject:Object) {
  if (my_FLVPlybk.contentPath != "http://www.helpexamples.com/flash/video/
  water.flv") {
    my_FLVPlybk.play("http://www.helpexamples.com/flash/video/water.flv");
  }
};
my_FLVPlybk.addEventListener("complete", listenerObject);
my_FLVPlybk.contentPath = "rtmp://my_servername/my_application/stream.flv";
```

Voir aussi

`FLVPlayback.activeVideoPlayerIndex`, `FLVPlayback.closeVideoPlayer()`, `FLVPlayback.contentPath`, `FLVPlayback.load()`, `FLVPlayback.play()`, `FLVPlayback.visibleVideoPlayerIndex`,

FLVPlayback.closeVideoPlayer()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
my_FLVPlybk.closeVideoPlayer(index:Number)
```

Paramètres

index Nombre correspondant à l'index du lecteur vidéo à fermer.

Renvoie

Objet VideoPlayer fermé.

Description

Méthode : ferme NetStream et supprime le lecteur vidéo spécifié par le paramètre *index*. Si le lecteur vidéo fermé est le lecteur vidéo actif ou visible, l'occurrence de composant FLVPlayback définit le lecteur vidéo actif et/ou visible sur le lecteur par défaut (avec l'index 0). Vous ne pouvez pas fermer le lecteur par défaut. Si vous essayez de le faire, le composant génère une erreur.

Exemple

L'exemple suivant crée deux lecteurs vidéo pour lire deux fichiers FLV de manière consécutive dans une même occurrence FLVPlayBack. Lorsque la lecture du second fichier FLV est terminée, le gestionnaire d'événements de l'événement `complete` appelle la méthode `closeVideoPlayer()` pour fermer le second lecteur. Si vous cliquez sur le bouton Lire pour lire les fichiers FLV une seconde fois, vous constatez l'absence du second lecteur vidéo, ce qui provoque la génération d'une erreur par le composant (VideoError) qui affiche un message indiquant que l'occurrence FLVPlayback ne peut pas trouver le fichier FLV.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
// Indication du nom et de l'emplacement du fichier FLV pour le
// lecteur par défaut.
```

```
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  clouds.flv";
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  // Ajout d'un second lecteur vidéo et indication du nom et
  // de l'emplacement de son fichier FLV.
  my_FLVPlybk.activeVideoPlayerIndex = 1;
  my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
  // Réinitialisation sur le lecteur vidéo par défaut qui lit
  // automatiquement son fichier FLV.
  my_FLVPlybk.activeVideoPlayerIndex = 0;
};
my_FLVPlybk.addEventListener("ready", listenerObject);
listenerObject.complete = function(eventObject:Object):Void {
  // Si l'événement complete concerne le second fichier FLV,
  // rendez-le actif et visible par défaut.
  if (eventObject.vp == 1) {
    my_FLVPlybk.activeVideoPlayerIndex = 0;
    my_FLVPlybk.visibleVideoPlayerIndex = 0;
    my_FLVPlybk.closeVideoPlayer(1);// Fermeture du second lecteur vidéo.
  } else { // Activation et affichage du second lecteur, lecture du fichier FLV.
    my_FLVPlybk.activeVideoPlayerIndex = 1;
    my_FLVPlybk.visibleVideoPlayerIndex = 1;
    my_FLVPlybk.play();
  }
};
// Ajout d'un écouteur pour un événement complete.
my_FLVPlybk.addEventListener("complete", listenerObject);
```

Voir aussi

FLVPlayback.close, FLVPlayback.activeVideoPlayerIndex, FLVPlayback.visibleVideoPlayerIndex

FLVPlayback.complete

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.complete = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("complete", listenerObject);
```

Description

Evénement : distribué à la fin de la lecture, car le lecteur a atteint la fin du fichier FLV. Le composant ne distribue pas l'événement si vous appelez les méthodes `stop()` ou `pause()` ou si vous cliquez sur les commandes correspondantes. L'objet événement a les propriétés `state` et `playheadTime`.

Lorsque l'application utilise le téléchargement progressif, ne définit pas la propriété `totalTime` de façon explicite et télécharge un fichier FLV qui n'indique pas la durée dans les métadonnées, le lecteur vidéo définit `totalTime` sur une valeur totale approximative avant de distribuer cet événement.

Le lecteur vidéo distribue également les événements `stateChange` et `stopped`.

Exemple

L'exemple suivant utilise la propriété `playheadTime` pour afficher la durée de lecture écoulée du fichier FLV dans le panneau Sortie lorsque l'événement `complete` se produit.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.complete = function(eventObject:Object):Void {
  trace("Elapsed play time at completion is: " + my_FLVPlybk.playheadTime);
};
my_FLVPlybk.addEventListener("complete", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.playheadTime`, `FLVPlayback.state`, `FLVPlayback.stateChange`, `FLVPlayback.stopped`, `FLVPlayback.totalTime`

FLVPlayback.CONNECTION_ERROR

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
mx.video.FLVPlayback.CONNECTION_ERROR
```

Description

Propriété de la classe FLVPlayback en lecture seule qui contient la constante de type chaîne « `connectionError` ». Vous pouvez comparer cette propriété avec la propriété `state` pour déterminer si une erreur de connexion s'est produite.

Exemple

L'exemple suivant force une erreur de connexion en indiquant un nom de fichier FLV non valide (`nosuch.flv`) dans la propriété `contentPath`. Cet exemple utilise la propriété `CONNECTION_ERROR` pour détecter l'erreur dans un écouteur d'événement `stateChange`.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  no_such.flv";
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  if(my_FLVPlybk.state == FLVPlayback.CONNECTION_ERROR)
    trace("State: " + FLVPlayback.CONNECTION_ERROR);
}
my_FLVPlybk.addEventListener("stateChange", listenerObject);
```

Voir aussi

`FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.contentPath

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.contentPath

Description

Propriété : chaîne indiquant l'URL du FLV à diffuser en flux continu et de quelle façon. L'URL peut être l'URL HTTP d'un fichier FLV, l'URL RTMP d'un flux ou l'URL HTTP d'un fichier XML.

Si vous définissez cette propriété dans l'inspecteur de composants ou l'inspecteur des propriétés, le chargement et la lecture du fichier FLV débutent à l'événement MovieClip.onEnterFrame suivant. Ce délai donne du temps pour définir les propriétés isLive, autoPlay et cuePoints, entre autres, qui affectent le chargement. Il permet également à ActionScript placé sur la première image d'avoir une incidence sur le composant FLVPlayback avant le début de la lecture.

Si vous définissez cette propriété via ActionScript, l'occurrence FLVPlayback ferme immédiatement le fichier FLV actuel et commence immédiatement à télécharger le nouveau fichier FLV. Les propriétés autoPlay, totalTime et isLive affectent le mode de chargement du nouveau fichier FLV. Ainsi, si vous les définissez, vous devez les définir *avant* la propriété contentPath.

Définissez la propriété autoPlay sur false pour empêcher la lecture automatique du nouveau fichier FLV.

Exemple

L'exemple suivant définit la propriété contentPath dans ActionScript pour indiquer l'emplacement du fichier FLV à lire.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. *N'affectez pas* de valeur au paramètre `contentPath` dans l'inspecteur de composants. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.metadataReceived = function(eventObject:Object):Void {
  my_FLVPlybk.setSize(my_FLVPlybk.preferredWidth,
  my_FLVPlybk.preferredHeight);
}
my_FLVPlybk.addEventListener("metadataReceived", listenerObject);
```

Voir aussi

`FLVPlayback.autoPlay`, `FLVPlayback.isLive`, `FLVPlayback.play()`, `FLVPlayback.totalTime`

FLVPlayback.cuePoint

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("cuePoint", listenerObject);
```

Description

Evénement ; distribué lorsqu'un point de repère est atteint. L'objet événement possède une propriété `info` qui contient l'objet `info` reçu par le rappel `NetStream.onCuePoint` pour les points de repère de fichier FLV. S'il s'agit de points de repère ActionScript, il contient l'objet transmis aux méthodes ou aux propriétés de points de repère ActionScript.

Cet événement a une propriété `vp`, qui correspond au numéro d'index du lecteur vidéo auquel cet événement s'applique.

Exemple

L'exemple suivant ajoute deux points de repère ActionScript à un fichier FLV. Le premier est ajouté à l'aide du paramètre *cuePoint* et le second en utilisant les paramètres *time* et *name*. Lorsque chaque point de repère se présente, un écouteur pour les événements `cuePoint` affiche la valeur de la propriété `playheadTime` dans une zone de texte.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Faites glisser un composant TextArea sur la scène, sous l'occurrence FLVPlayback, et nommez l'occurrence, TextArea **my_ta**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
  - Composant TextArea sur la scène dont l'occurrence est nommée my_ta
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
my_ta.visible = false;
// Création d'un objet point de repère.
var cuePt:Object = new Object(); // Création d'un objet point de repère.
cuePt.time = 1.444;
cuePt.name = "elapsed_time";
cuePt.type = "actionscript";
my_FLVPlybk.addASCuePoint(cuePt);  // Ajout d'un point de repère AS.
// Ajout du second point de repère AS à l'aide des paramètres d'heure et de nom.
my_FLVPlybk.addASCuePoint(5.3, "elapsed_time2");
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object):Void {
    my_ta.text = "Cue at: " + eventObject.info.time + " occurred";
    my_ta.visible = true;
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
```

Voir aussi

`FLVPlayback.activeVideoPlayerIndex`, `FLVPlayback.visibleVideoPlayerIndex`

FLVPlayback.cuePoints

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.cuePoints

Description

Propriété : tableau décrivant des points de repère ActionScript et des points de repère intégrés au fichier FLV désactivés. Cette propriété est créée spécialement pour être utilisée par l'inspecteur de composants. Elle ne fonctionne pas si elle est définie autrement. Sa valeur a une incidence uniquement sur le premier fichier FLV chargé, et seulement s'il est chargé en définissant la propriété `contentPath` dans l'inspecteur de composants ou l'inspecteur des propriétés.

REMARQUE	Cette propriété n'est pas accessible dans ActionScript. Pour accéder aux informations sur les points de repère dans ActionScript, utilisez la propriété `metadata`.

Pour ajouter, supprimer, activer ou désactiver des points de repère avec ActionScript, utilisez `addASCuePoint()`, `removeASCuePoint()` ou `setFLVCuePointEnabled()`.

Voir aussi

`FLVPlayback.contentPath`, `FLVPlayback.addASCuePoint()`, `FLVPlayback.findCuePoint()`, `FLVPlayback.findNearestCuePoint()`, `FLVPlayback.findNextCuePointWithName()`, `FLVPlayback.isFLVCuePointEnabled()`, `FLVPlayback.metadata`, `FLVPlayback.metadataReceived`, `FLVPlayback.removeASCuePoint()`, `FLVPlayback.seekToNavCuePoint()`, `FLVPlayback.seekToNextNavCuePoint()`, `FLVPlayback.seekToPrevNavCuePoint()`, `FLVPlayback.setFLVCuePointEnabled()`

FLVPlayback.DISCONNECTED

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
mx.video.FLVPlayback.DISCONNECTED
```

Description

Propriété de la classe FLVPlayback en lecture seule qui contient la constante de type chaîne « `disconnected` ». Vous pouvez comparer cette propriété avec la propriété `state` pour déterminer si un état déconnecté existe.

L'occurrence FLVPlayback est en état déconnecté tant que vous n'avez pas défini la propriété `contentPath`.

Exemple

L'exemple suivant affiche simplement un message dans le panneau Sortie qui confirme que l'occurrence FLVPlayback est en état déconnecté avant de définir la propriété `contentPath`.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;

if(my_FLVPlybk.state == FLVPlayback.DISCONNECTED)
  trace("FLVPlayback instance is currently disconnected");
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.contentPath`, `FLVPlayback.state`

FLVPlayback.EVENT

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`mx.video.FLVPlayback.EVENT`

Description

Propriété de la classe FLVPlayback en lecture seule qui contient la constante de type chaîne « `event` ». Vous pouvez utiliser cette propriété comme paramètre `type` pour les méthodes `findCuePoint()` et `findNearestCuePoint()`.

Exemple

L'exemple suivant utilise la propriété `FLVPlayback.EVENT` pour indiquer qu'il recherche un point de repère nommé `myCue` de type `event`. Il affiche les propriétés `name`, `time` et `type` du point de repère renvoyé, et l'écouteur `cuePoint` affiche « Hit it! » dans le panneau Sortie lorsque le point de repère se présente.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  plane_cuepoints.flv";
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  if(rtn_cuePt = my_FLVPlybk.findCuePoint("myCue", FLVPlayback.EVENT)){
    trace("Cue point name is: " + rtn_cuePt.name);
    trace("Cue point time is: " + rtn_cuePt.time);
    trace("Cue point type is: " + rtn_cuePt.type);
  }
}
```

```
my_FLVPlybk.addEventListener("ready", listenerObject);
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object):Void {
  if(eventObject.info.name == "myCue")
    trace("Hit it!");
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
```

Voir aussi

`FLVPlayback.findCuePoint()`

FLVPlayback.fastForward

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.fastForward = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("fastForward", listenerObject);
```

Evénement ; distribué lorsque la tête de lecture avance au cours de la recherche, manuellement, via ActionScript ou lorsque vous cliquez sur le bouton ForwardButton. L'objet événement a les propriétés `state`, `playheadTime` et `vp`. La propriété `playheadTime` reflète l'heure de destination alors que la propriété `vp` correspond au numéro d'index du lecteur vidéo auquel l'événement s'applique.

L'occurrence FLVPlayback distribue également les événements `seek` et `playheadUpdate`.

Exemple

L'exemple suivant utilise des occurrences de l'événement `fastForward` lorsqu'il a lieu et affiche la durée de lecture écoulée dans le panneau Sortie. Lorsque l'événement `ready` se produit, un appel à la méthode `seekPercent()` déclenche l'évenement `fastForward`.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  my_FLVPlybk.seekPercent(35);
};
my_FLVPlybk.addEventListener("ready", listenerObject);

listenerObject.fastForward = function(eventObject:Object):Void {
  trace("fastforward event; playhead time is: " +
  eventObject.playheadTime);
};
my_FLVPlybk.addEventListener("fastForward", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

FLVPlayback.activeVideoPlayerIndex, FLVPlayback.forwardButton, FLVPlayback.seek, FLVPlayback.seek(), FLVPlayback.seekBar, FLVPlayback.seekPercent(), FLVPlayback.seekSeconds(), FLVPlayback.seekToNavCuePoint(), FLVPlayback.seekToNextNavCuePoint(), FLVPlayback.seekToPrevNavCuePoint(), FLVPlayback.seekToPrevOffset, FLVPlayback.state, FLVPlayback.playheadTime

FLVPlayback.findCuePoint()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
my_FLVplybk.findCuePoint(time:Number[, type:String]):Object
my_FLVplybk.findCuePoint(name:String[, type:String]):Object
my_FLVplybk.findCuePoint(cuePoint:Object[, type:String]):Object
```

Paramètres

time Nombre correspondant à l'heure, en secondes, du point de repère à rechercher. La méthode utilise uniquement les trois premiers chiffres et arrondit tout chiffre supplémentaire indiqué. Elle renvoie le point de repère qui correspond à cette heure. Si plusieurs points de repère ActionScript ont la même heure, la méthode renvoie uniquement celui dont le nom est le premier dans l'ordre alphabétique. Si elle ne trouve pas de correspondance, elle renvoie la valeur `null`.

name Chaîne indiquant le nom du point de repère à rechercher. La méthode renvoie le premier point de repère qui correspond à ce nom ou la valeur `null` si elle ne trouve pas de correspondance.

cuePoint Objet point de repère qui contient les propriétés `time` et `name` en fonction desquelles effectuer la recherche. Si la propriété `name` n'a pas de valeur ou la valeur `null`, la recherche se comporte comme si le paramètre était un nombre représentant l'heure à rechercher. Si la propriété `time` n'a pas de valeur, la valeur `null` ou une valeur négative, la recherche se comporte comme si le paramètre était une chaîne contenant le nom à rechercher. Si vous indiquez les deux propriétés `time` et `name` et qu'un point de repère correspondant à ces valeurs existe, la méthode le renvoie. Sinon, la méthode renvoie la valeur `null`.

type Facultatif. Chaîne qui indique le type de point de repère à rechercher. Les valeurs possibles de ce paramètre sont les suivantes : « `actionscript` », « `all` », « `event` », « `flv` » ou « `navigation` ». Vous pouvez spécifier ces valeurs à l'aide des propriétés de classe suivantes : `FLVPlayback.ACTIONSCRIPT`, `FLVPlayback.ALL`, `FLVPlayback.EVENT`, `FLVPlayback.FLV` et `FLVPlayback.NAVIGATION`. Si ce paramètre n'est pas précisé, sa valeur par défaut est « `all` », ce qui signifie que la méthode recherche tous les types de points de repère.

Renvoie

Objet qui est une copie de l'objet point de repère avec les propriétés supplémentaires suivantes :

`array` Tableau des points de repère cherchés. Considérez ce tableau comme étant en lecture seule, car si vous y ajoutez, supprimez ou modifiez des objets, les points de repère risquent de ne pas fonctionner correctement.

`index` Index dans le tableau pour le point de repère renvoyé.

Renvoie la valeur `null` si aucune correspondance n'est trouvée.

Description

Méthode : recherche le point de repère du type spécifié par le paramètre *type* et ayant l'heure, le nom ou l'heure et le nom que vous avez indiqués via les paramètres.

Si vous n'indiquez pas de valeur pour l'heure ou le nom du point de repère, ou si l'heure a la valeur `null`, si elle n'est pas définie ou d'une valeur négative *et* si le nom a la valeur `null` ou s'il n'est pas défini, la méthode génère l'erreur VideoError 1002. Pour plus d'informations, reportez-vous à « Classe VideoError », à la page 723.

Cette méthode inclut les points de repère désactivés dans la recherche. Utilisez la méthode `isFLVCuePointEnabled()` pour déterminer si un point de repère est désactivé.

Exemple

L'exemple suivant ajoute deux points de repère ActionScript dans un fichier FLV, puis appelle la méthode `findCuePoint()` trois fois. Le premier appel recherche un point de repère de navigation ayant 7.748 comme heure. Le deuxième appel recherche le point de repère « `ASCue1` », en utilisant une seule valeur de nom. Le troisième appel utilise un objet point de repère indiquant à la fois une heure, `10`, et un nom, « `ASCue2` ». Après ce troisième appel, l'exemple affiche le contenu de l'objet point de repère renvoyé, notamment le tableau de points de repère recherchés. Dans cet exemple, il s'agit de points de repère ActionScript.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
// Création d'un objet point de repère.
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
var cuePt:Object = new Object();
var rtn_cuePt:Object = new Object(); // Création d'un objet pour la
                                     // valeur renvoyée.
cuePt.time = 4.444;
cuePt.name = "AScue1";
my_FLVPlybk.addASCuePoint(cuePt);  // Ajout d'un point de repère AS.
// Ajout du second point de repère AS à l'aide des paramètres d'heure et de nom.
my_FLVPlybk.addASCuePoint(10, "AScue2");
// Recherche d'un point de repère de navigation en utilisant l'heure.
rtn_cuePt = my_FLVPlybk.findCuePoint(7.748, FLVPlayback.NAVIGATION);
// Recherche d'un point de repère AS en utilisant uniquement le nom.
rtn_cuePt = my_FLVPlybk.findCuePoint("AScue1");
// Recherche d'un point de repère AS en utilisant l'objet point de repère.
cuePt.time = 10;
cuePt.name = "AScue2";
rtn_cuePt = my_FLVPlybk.findCuePoint(cuePt, FLVPlayback.ACTIONSCRIPT);
```

```
// Vérification du contenu de l'objet point de repère renvoyé.
for (i in rtn_cuePt) {
  // S'il s'agit d'un objet tableau, ouvrez-le et suivez son contenu.
  if (typeof rtn_cuePt[i] == "object") {
    tracer(rtn_cuePt[i]);
  // Suivi de la paire nom (i) et valeur (rtn_cuePt[i]).
  } else {
    trace(i+ " " + rtn_cuePt[i]);
  }
}
// Suivi du tableau des points de repère.
function tracer(cuepts:Array) {
  for (i in cuepts) {
    if (typeof cuepts[i] == "object") { // Si objet dans objet,
        tracer(cuepts[i]); // suivi de l'objet,
        } else {
       // suivi du nom : valeur
        trace(i + " " + cuepts[i]);
        }
  }
}
```

Voir aussi

FLVPlayback.addASCuePoint(), FLVPlayback.cuePoints, FLVPlayback.findNearestCuePoint(), FLVPlayback.findNextCuePointWithName(), FLVPlayback.isFLVCuePointEnabled(), FLVPlayback.removeASCuePoint(), FLVPlayback.seekToNavCuePoint(), FLVPlayback.seekToNextNavCuePoint(), FLVPlayback.seekToPrevNavCuePoint(), FLVPlayback.setFLVCuePointEnabled()

FLVPlayback.findNearestCuePoint()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
my_FLVplybk.findNearestCuePoint(time:Number[, type:String]):Object
my_FLVplybk.findNearestCuePoint(name:String[, type:String]):Object
my_FLVplybk.findNearestCuePoint(cuePoint:Object[, type:String]):Object
```

Paramètres

time Nombre correspondant à l'heure, en secondes, du point de repère à rechercher. La méthode utilise uniquement les trois premiers chiffres et arrondit tout chiffre supplémentaire indiqué. Elle renvoie le point de repère qui correspond à cette heure ou le point de repère *antérieur* le plus proche, du type spécifié. Si plusieurs points de repère ont la même heure, ce qui ne peut se produire qu'avec des points de repère ActionScript, la méthode renvoie uniquement celui dont le nom est le premier dans l'ordre alphabétique. Si elle ne trouve pas de correspondance, elle renvoie la valeur `null`.

name Chaîne indiquant le nom du point de repère à rechercher. La méthode renvoie le premier point de repère qui correspond à ce nom ou la valeur `null` si elle ne trouve pas de correspondance.

cuePoint Objet point de repère qui contient les propriétés `time` et `name` en fonction desquelles effectuer la recherche. Si vous indiquez une heure et que la propriété `name` n'a pas de valeur ou la valeur `null`, la recherche se comporte comme si le paramètre était un nombre représentant l'heure à rechercher. Si vous indiquez un nom et que la propriété `time` n'a pas de valeur, la valeur `null` ou une valeur négative, la recherche se comporte comme si le paramètre était une chaîne contenant le nom à rechercher. Si vous indiquez les deux propriétés `time` et `name` et qu'un point de repère correspondant à ces valeurs existe, la méthode le renvoie. Si elle ne trouve pas de correspondance pour à la fois l'heure et le nom, elle renvoie le premier point de repère correspondant au nom et ayant une heure *antérieure*. S'il n'existe aucun point de repère antérieur portant ce nom, elle renvoie le premier point de repère qui correspond à ce nom. Sinon, la méthode renvoie la valeur `null`.

type Facultatif. Chaîne qui indique le type de point de repère à rechercher. Les valeurs possibles de ce paramètre sont les suivantes : « `actionscript` », « `all` », « `event` », « `flv` » ou « `navigation` ». Vous pouvez spécifier ces valeurs à l'aide des propriétés de classe suivantes : `FLVPlayback.ACTIONSCRIPT`, `FLVPlayback.ALL`, `FLVPlayback.EVENT`, `FLVPlayback.FLV` et `FLVPlayback.NAVIGATION`. Si ce paramètre n'est pas spécifié, la valeur par défaut est « `all` », ce qui signifie que la méthode recherche tous les types de points de repère.

Renvoie

Objet qui est une copie de l'objet point de repère trouvé avec les propriétés supplémentaires suivantes :

`array` Tableau des points de repère cherchés. Considérez ce tableau comme étant en lecture seule, car si vous y ajoutez, supprimez ou modifiez des objets, les points de repère risquent de ne pas fonctionner correctement.

`index` Index situé dans le tableau du point de repère renvoyé.

Renvoie la valeur `null` si aucune correspondance n'est trouvée.

Description

Méthode : recherche un point de repère du type spécifié qui correspond *ou est antérieur* à l'heure indiquée, ou qui correspond au nom spécifié, si vous indiquez à la fois une heure et un nom et qu'aucun point de repère antérieur ne porte ce nom. Sinon, elle renvoie la valeur `null`.

Cette méthode inclut les points de repère désactivés dans la recherche. Utilisez la méthode `isFLVCuePointEnabled()` pour déterminer si un point de repère est désactivé.

Si l'heure a la valeur `null`, si elle n'est pas définie ou si sa valeur est négative *et* que le nom a la valeur `null` ou qu'il n'est pas défini, la méthode génère l'erreur VideoError 1002.

Exemple

L'exemple suivant crée un point de repère ActionScript pour le fichier FLV à 4.07 secondes. Lorsque ce point de repère survient, le gestionnaire d'événements `cuePoint` appelle la méthode `findNearestCuePoint()` pour trouver un point de repère d'un type quelconque situé 5 secondes après au plus proche. Le panneau Sortie affiche le nom, l'heure et le type du point de repère renvoyé.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
var cuePt:Object = new Object(); // Création d'un objet point de repère.
cuePt.time = 4.07;
cuePt.name = "ASpt1";
cuePt.type = "actionscript";
my_FLVPlybk.addASCuePoint(cuePt);  // Ajout d'un point de repère AS.
function cuePoint(eventObject:Object):Void {
  if (eventObject.info.name == "ASpt1") {
    var rtn_obj:Object = new Object();
    rtn_obj = my_FLVPlybk.findNearestCuePoint(eventObject.info.time + 5);
    trace("Cue point name is: " + rtn_obj.name);
    trace("Cue point time is: " + rtn_obj.time);
    trace("Cue point type is: " + rtn_obj.type);
  }
}
my_FLVPlybk.addEventListener("cuePoint", cuePoint);
```

Voir aussi

`FLVPlayback.addASCuePoint()`, `FLVPlayback.cuePoints`, `FLVPlayback.findCuePoint()`, `FLVPlayback.findNextCuePointWithName()`, `FLVPlayback.isFLVCuePointEnabled()`, `FLVPlayback.removeASCuePoint()`, `FLVPlayback.seekToNavCuePoint()`, `FLVPlayback.seekToNextNavCuePoint()`, `FLVPlayback.seekToPrevNavCuePoint()`, `FLVPlayback.setFLVCuePointEnabled()`

FLVPlayback.findNextCuePointWithName()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVplybk.findNextCuePointWithName(*my_cuePoint*:Object)

Paramètres

mon_point_repère Objet point de repère renvoyé par la méthode `findCuePoint()`, la méthode `findNearestCuePoint()` ou un précédent appel à cette méthode.

Renvoie

Objet qui est une copie de l'objet point de repère trouvé avec les propriétés supplémentaires suivantes :

`array` Tableau des points de repère cherchés. Considérez ce tableau comme étant en lecture seule, car si vous y ajoutez, supprimez ou modifiez des objets, les points de repère risquent de ne pas fonctionner correctement.

`index` Index situé dans le tableau du point de repère renvoyé.

Renvoie la valeur `null` si aucune correspondance n'est trouvée.

Description

Méthode : recherche le point de repère suivant dans *mon_point_repère*.`array` portant le même nom que *mon_point_repère*.`name`. L'objet *mon_point_repère* doit être un objet point de repère renvoyé par la méthode `findCuePoint()`, la méthode `findNearestCuePoint()` ou un précédent appel à cette méthode. Cette méthode utilise la propriété `array` que ces méthodes ajoutent à l'objet point de repère.

Cette méthode inclut les points de repère désactivés dans la recherche. Utilisez la méthode `isFLVCuePointEnabled()` pour déterminer si un point de repère est désactivé.

Renvoie la valeur `null` si le tableau ne contient plus de points de repère ayant un nom correspondant.

Exemple

L'exemple suivant crée trois points de repère ActionScript portant le nom « `transition` ». Lorsque l'événement `ready` se produit, le gestionnaire d'événements appelle la méthode `findCuePoint()` pour rechercher le premier point de repère portant ce nom. Si elle trouve une correspondance, elle appelle la fonction `findNextName()`, qui appelle la méthode `findNextCuePointWithName()`, en transmettant l'objet point de repère renvoyé (`rtn_obj`) afin de rechercher d'autres points de repère avec le même nom.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
var cuePt:Object = new Object(); // Création d'un objet point de repère.
cuePt.time = 6.27;
cuePt.name = "transition";
cuePt.type = "actionscript";
my_FLVPlybk.addASCuePoint(cuePt);  // Ajout d'un point de repère AS.
cuePt.time = 7.06;
my_FLVPlybk.addASCuePoint(cuePt);  // Ajout d'un point de repère AS.
cuePt.time = 11.13;
my_FLVPlybk.addASCuePoint(cuePt);  // Ajout d'un point de repère AS.
var listenerObject:Object = new Object();
listenerObject.ready = function():Void {
  var rtn_obj:Object = new Object();
// Recherche d'un point de repère AS en utilisant uniquement le nom.
  if (rtn_obj = my_FLVPlybk.findCuePoint("transition")) {
    trace("Cue point name is: " + rtn_obj.name);
    trace("Cue point time is: " + rtn_obj.time);
    trace("Cue point type is: " + rtn_obj.type);
    findNextName(rtn_obj);
  }
}
my_FLVPlybk.addEventListener("ready", listenerObject);
// Recherche d'autres points de repère portant le même nom.
function findNextName(cuePt:Object):Void {
  while(cuePt = my_FLVPlybk.findNextCuePointWithName(cuePt)) {
       trace("Cue point name is: " + cuePt.name);
       trace("Cue point time is: " + cuePt.time);
       trace("Cue point type is: " + cuePt.type);
  }
}
```

Voir aussi

FLVPlayback.addASCuePoint(), FLVPlayback.cuePoints, FLVPlayback.findCuePoint(), FLVPlayback.findNearestCuePoint(), FLVPlayback.isFLVCuePointEnabled(), FLVPlayback.removeASCuePoint(), FLVPlayback.seekToNavCuePoint(), FLVPlayback.seekToNextNavCuePoint(), FLVPlayback.seekToPrevNavCuePoint(), FLVPlayback.setFLVCuePointEnabled()

FLVPlayback.FLV

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`mx.video.FLVPlayback.FLV`

Description

Propriété de la classe FLVPlayback en lecture seule qui contient la constante de type chaîne « `flv` ». Vous pouvez utiliser cette propriété comme paramètre `type` des méthodes `findCuePoint()` et `findNearestCuePoint()`.

Exemple

L'exemple suivant recherche un point de repère nommé `point2` avec une heure égale à 7.748 parmi des points de repère de fichier FLV, puis affiche le type et l'heure trouvés. Les points de repère de fichier FLV sont des points de repère de navigation et d'événement.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
// Création d'un objet point de repère.
var listenerObject = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  var cuePt:Object = new Object(); // Création d'un objet point de repère.
  cuePt.name = "point3";
  cuePt.time = 16.02;
  if(cuePt = my_FLVPlybk.findCuePoint(cuePt, FLVPlayback.FLV)) // Recherche
                                                  // d'un point de repère.
    trace("found a " + cuePt.type + " cue point at " + cuePt.time);
  else
    trace("cue point not found");
}
my_FLVPlybk.addEventListener("ready", listenerObject);
```

Voir aussi

`FLVPlayback.findCuePoint()`, `FLVPlayback.findNearestCuePoint()`

FLVPlayback.forwardButton

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.forwardButton

Description

Propriété : objet MovieClip qui correspond à la commande du bouton Avance. Pour plus d'informations sur l'utilisation des composants de l'interface utilisateur personnalisée Lecture de fichiers FLV pour les commandes de lecture, reportez-vous à « Application d'enveloppe aux composants particuliers de l'interface utilisateur personnalisée Lecture de fichiers FLV », à la page 544.

Exemple

L'exemple suivant utilise les propriétés `backButton`, `forwardButton`, `playButton`, `pauseButton` et `stopButton` pour associer des commandes individuelles de l'interface utilisateur personnalisée Lecture de fichiers FLV à un composant FLVPlayback.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Dans l'inspecteur des composants, définissez le paramètre Skin (Enveloppe) sur None. Ajoutez ensuite les composants individuels suivants de l'interface utilisateur personnalisée FLV et donnez-leur les noms d'occurrence indiqués entre parenthèses : BackButton (**my_bkbttn**), ForwardButton (**my_fwdbttn**), PlayButton (**my_plybttn**), PauseButton (**my_pausbttn**) et StopButton (**my_stopbttn**). Ajoutez ensuite les lignes de code suivantes au panneau Actions :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
  - Composants BackButton, ForwardButton, PlayButton, PauseButton et
   StopButton de l'interface utilisateur personnalisée FLV dans la
  bibliothèque
*/
import mx.video.*;
my_FLVPlybk.backButton = my_bkbttn;
my_FLVPlybk.forwardButton = my_fwdbttn;
my_FLVPlybk.playButton = my_plybttn;
```

```
my_FLVPlybk.pauseButton = my_pausbttn;
my_FLVPlybk.stopButton = my_stopbttn;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.fastForward`, `FLVPlayback.seek`, `FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.getVideoPlayer()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`my_FLVplybk``.getVideoPlayer(`*`index`*`:Number)`

Renvoie

Objet VideoPlayer.

Description

Méthode ; obtient le lecteur vidéo spécifié par le paramètre *`index`*. Dans la mesure du possible, il est préférable d'accéder aux méthodes et aux propriétés VideoPlayer à l'aide des méthodes et des propriétés FLVPlayback. Chaque propriété `VideoPlayer._name` est son index.

Exemple

L'exemple suivant utilise deux lecteurs vidéo pour lire deux fichiers FLV. Lorsque le second fichier FLV déclenche l'événement `ready`, l'exemple appelle la méthode `getVideoPlayer()` pour obtenir le lecteur vidéo numéro 1, puis définit sa propriété `_alpha` sur 50. Cette procédure entraîne la transparence du fichier FLV (plane_cuepoints) dans ce lecteur, puis affiche simultanément les deux fichiers FLV.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
my_FLVPlybk.load("http://www.helpexamples.com/flash/video/cuepoints.flv");
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object) {
    if (eventObject.target.contentPath == "http://www.helpexamples.com/
  flash/video/cuepoints.flv") {
        // Déclenchement une fois que le premier fichier FLV est prêt.
        my_FLVPlybk.activeVideoPlayerIndex = 1;
        my_FLVPlybk.load("http://www.helpexamples.com/flash/video/
  plane_cuepoints.flv");
    } else {
        // Déclenchement une fois que le second fichier FLV est prêt.
        eventObject.target.activeVideoPlayerIndex = 0;
        eventObject.target.play();
        eventObject.target.activeVideoPlayerIndex = 1;
        eventObject.target.play();
        var layerOnTop:MovieClip = eventObject.target.getVideoPlayer(1);
        layerOnTop._alpha = 50;
        layerOnTop._visible = true;
    }
}
my_FLVPlybk.addEventListener("ready", listenerObject);
```

Voir aussi

FLVPlayback.activeVideoPlayerIndex, FLVPlayback.bringVideoPlayerToFront(), FLVPlayback.visibleVideoPlayerIndex

FLVPlayback.height

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.height

Description

Propriété : nombre indiquant la hauteur de l'occurrence FLVPlayback. Cette propriété affecte uniquement la hauteur de l'occurrence FLVPlayback et n'inclut pas la hauteur d'un fichier SWF d'enveloppe qui peut être chargé. Utilisez la propriété `height` de l'occurrence FLVPlayback, mais pas la propriété `MovieClip._height`, car la propriété `_height` risque de donner une autre valeur en cas de chargement d'un fichier SWF d'enveloppe.

Exemple

L'exemple suivant définit les propriétés `width` et `height` afin de modifier la taille du lecteur vidéo. Il définit d'abord la propriété `maintainAspectRatio` sur `false` pour empêcher le redimensionnement automatique du lecteur vidéo en cas de changement des dimensions.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.maintainAspectRatio = false;
my_FLVPlybk.width = 300;
my_FLVPlybk.height = 350;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.preferredHeight`, `FLVPlayback.preferredWidth`, `FLVPlayback.maintainAspectRatio`, `FLVPlayback.resize`, `FLVPlayback.setSize()`, `FLVPlayback.width`

FLVPlayback.idleTimeout

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.idleTimeout

Description

Propriété : temps écoulé (en millisecondes) avant que Flash mette fin à une connexion inactive au serveur FMS en raison de l'interruption ou de l'arrêt de la lecture. Cette propriété n'a aucune incidence sur le téléchargement HTTP d'un fichier FLV.

Si cette propriété est définie lorsqu'un flux vidéo est déjà inactif, elle redémarre la période de délai d'attente avec la nouvelle valeur.

La valeur par défaut est 300 000 (5 minutes).

Exemple

L'exemple suivant suppose que vous lisez un fichier FLV en flux continu à partir d'un FMS ou de FVSS. Cet exemple définit la propriété `idleTimeout` sur une valeur faible égale à 10 millisecondes, qui déclenche un dépassement de délai, et en conséquence, un événement `close` sur la connexion RTMP. L'écouteur d'événement `close` affiche le numéro d'index du lecteur vidéo pour lequel l'événement s'est produit.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Dans l'inspecteur de composants, affectez au paramètre `contentPath` une valeur indiquant l'emplacement d'un fichier FLV diffusé en flux continu à partir d'un FMS ou FVSS. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.idleTimeout = 10;
var listenerObject:Object = new Object();
// Ecoute de l'événement close sur la connexion RTMP ; affichage
// de l'index du lecteur vidéo.
listenerObject.close = function(eventObject:Object) {
  trace("Closed connection for video player: " + eventObject.vp);
};
my_FLVPlybk.addEventListener("close", listenerObject);
```

Voir aussi

`FLVPlayback.close`

FLVPlayback.isFLVCuePointEnabled()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
my_FLVplybk.isFLVCuePointEnabled(time:Number)
my_FLVplybk.isFLVCuePointEnabled(name:String)
my_FLVplybk.isFLVCuePointEnabled(cuePoint:Object)
```

Paramètres

time Nombre correspondant à l'heure, en secondes, du point de repère à rechercher.

name Chaîne indiquant le nom du point de repère à rechercher.

cuePoint Objet point de repère possédant les propriétés `time` et `name` correspondant à ce point de repère. Cette méthode n'active aucune autre propriété sur l'objet point de repère entrant. Si la propriété `time` ou `name` est définie sur undefined, la méthode utilise uniquement la propriété qui est définie.

Renvoie

Valeur booléenne définie sur `false` si le ou les points de repère sont trouvés et désactivés, et sur `true` s'ils ne sont pas désactivés ou s'ils n'existent pas. Si l'heure donnée n'est pas définie, si sa valeur est `null`, négative ou si un seul point de repère est fourni, la méthode renvoie la valeur `false` uniquement si tous les points de repère portant ce nom sont désactivés.

Description

Méthode : renvoie la valeur `false` si le point de repère intégré au fichier FLV est désactivé. Vous pouvez désactiver les points de repère en définissant la propriété `cuePoints` dans la boîte de dialogue Points de repère des vidéos Flash ou en appelant la méthode `setFLVCuePointEnabled()`.

La valeur renvoyée par cette fonction est significative uniquement si la propriété `metadataLoaded` est définie sur `true`, si la propriété `metadata` n'est pas `null` ou après un événement `metadataReceived`. Si la propriété `metadataLoaded` est définie sur `false`, cette fonction renvoie toujours la valeur `true`.

Exemple

L'exemple suivant désactive le point de repère `point2` lorsque l'événement `ready` se produit. Lorsque le premier événement `cuePoint` se produit, le gestionnaire d'événements appelle la méthode `isFLVCuePointEnabled()` pour vérifier si le point de repère est désactivé. Le cas échéant, il l'active. Le fichier FLV contient les points de repère intégrés suivants : point1, 00:00:00:418 ; point2, 00:00:07.748 ; point3, 00:00:16:020.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
function ready(eventObject:Object) {
  my_FLVPlybk.setFLVCuePointEnabled(false, "point2");
}
my_FLVPlybk.addEventListener("ready", ready);
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object) {
    trace("Elapsed time in seconds: " + my_FLVPlybk.playheadTime);
    trace("Cue point name is: " + eventObject.info.name);
    trace("Cue point type is: " + eventObject.info.type);
    if (my_FLVPlybk.isFLVCuePointEnabled("point2") == false) {
      my_FLVPlybk.setFLVCuePointEnabled(true, "point2");
    }
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Voir aussi

`FLVPlayback.cuePoint`, `FLVPlayback.findCuePoint()`, `FLVPlayback.findNearestCuePoint()`, `FLVPlayback.findNextCuePointWithName()`, `FLVPlayback.setFLVCuePointEnabled()`, `FLVPlayback.seekToNavCuePoint()`, `FLVPlayback.seekToNextNavCuePoint()`, `FLVPlayback.seekToPrevNavCuePoint()`

FLVPlayback.isLive

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.isLive

Description

Propriété ; valeur booléenne définie sur `true` si le flux vidéo est en direct. Cette propriété est significative uniquement lors de la diffusion en continu à partir d'un FMS ou FVSS. La valeur de cette propriété est ignorée pour un téléchargement HTTP.

Si vous définissez cette propriété entre le chargement de nouveaux fichiers FLV, elle n'a aucune incidence tant que le paramètre `contentPath` n'est pas défini pour le nouveau fichier FLV.

Exemple

L'exemple suivant suppose que vous lisez un flux en direct à partir d'un FMS. Lorsque l'événement `playing` se produit, l'exemple affiche la valeur de la propriété `isLive`.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez ensuite le code suivant dans le panneau Actions, sur l'image 1 du scénario. Dans l'instruction qui charge la propriété `contentPath`, remplacez le texte en italique par le nom et l'emplacement d'un fichier FLV sur votre serveur FMS.

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.playing = function(eventObject:Object) {
  trace("The isLive property is " + my_FLVPlybk.isLive);
};
my_FLVPlybk.addEventListener("playing", listenerObject);
my_FLVPlybk.contentPath = "rtmp://my_servername/my_application/stream.flv";
```

Voir aussi

`FLVPlayback.contentPath`, `FLVPlayback.load()`

FLVPlayback.isRTMP

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.isRTMP

Description

Propriété : valeur booléenne définie sur `true` si le fichier FLV est diffusé en continu à partir d'un serveur FMS ou FVSS via RTMP. Sa valeur est définie sur `false` pour toute autre source de fichier FLV. Lecture seule.

Exemple

L'exemple suivant suppose que vous lisez un fichier FLV en flux continu à partir d'un FMS ou de FVSS. Lorsque l'événement `playing` se produit, l'exemple affiche la valeur de la propriété `isRTMP` pour indiquer si le fichier FLV provient d'une URL RTMP.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez ensuite le code suivant dans le panneau Actions, sur l'image 1 du scénario. Dans l'instruction qui charge la propriété `contentPath`, remplacez le texte en italique par le nom et l'emplacement d'un fichier FLV sur votre serveur FMS.

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.bufferTime = 7;
var listenerObject:Object = new Object();
// Ecoute de l'événement playing sur la connexion RTMP ;
// affichage du résultat isRTMP.
listenerObject.playing = function(eventObject:Object) {
  trace("Value of isRTMP property is: " + my_FLVPlybk.isRTMP);
};
my_FLVPlybk.addEventListener("playing", listenerObject);
my_FLVPlybk.contentPath = "rtmp://my_servername/my_application/stream.flv";
```

Voir aussi

`FLVPlayback.contentPath`, `FLVPlayback.load()`, `FLVPlayback.play()`

FLVPlayback.load()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVplybk.load(*contentPath*:String[, *totalTime*:Number, *isLive*:Boolean])

Paramètres

contentPath Chaîne indiquant l'URL du fichier FLV à diffuser en continu et de quelle façon. L'URL peut être un chemin local, l'URL HTTP d'un fichier FLV, l'URL RTMP d'un flux de fichier FLV ou l'URL HTTP d'un fichier XML.

totalTime Nombre représentant la durée de lecture totale de la vidéo. Facultatif.

isLive Valeur booléenne définie sur `true` si le flux vidéo est en direct. Cette valeur est significative uniquement lors de la diffusion en continu à partir de FVSS ou FMS. La valeur de cette propriété est ignorée s'il s'agit d'un téléchargement HTTP. Valeur facultative.

Renvoie

Aucune.

Description

Méthode : démarre le chargement du fichier FLV et fournit un raccourci pour définir la propriété `autoPlay` sur `false` et les propriétés `contentPath`, `totalTime` et `isLive`, le cas échéant. Si les propriétés `totalTime` et `isLive` ont la valeur undefined, elles ne sont pas définies. Si la propriété `contentPath` n'est pas définie, a une valeur `null` ou s'il s'agit d'une chaîne vide, cette méthode n'effectue aucune action.

Exemple

L'exemple suivant appelle la méthode `load()` pour charger un fichier FLV spécifié par le paramètre `contentPath`. Il affiche la valeur de la propriété `autoPlay` avant et après le chargement de ce fichier et appelle la méthode `play()` pour commencer à le lire.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
trace("Before load, autoPlay is: " + my_FLVPlybk.autoPlay);
my_FLVPlybk.load("http://www.helpexamples.com/flash/video/water.flv");
trace("After load, autoPlay is: " + my_FLVPlybk.autoPlay);
my_FLVPlybk.play();
```

Voir aussi

`FLVPlayback.contentPath`, `FLVPlayback.isLive`, `FLVPlayback.totalTime`

FLVPlayback.LOADING

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
mx.video.FLVPlayback.LOADING
```

Description

Propriété de la classe FLVPlayback en lecture seule qui contient la constante de type chaîne « `loading` ». Vous pouvez comparer cette propriété avec la propriété `state` pour déterminer si le composant est en état de chargement.

Exemple

L'exemple suivant affiche la valeur de la propriété `FLVPlayback.LOADING` si le fichier FLV est en état de chargement.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;

var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  if(my_FLVPlybk.state == FLVPlayback.LOADING)
    trace("State is " + FLVPlayback.LOADING);
}
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.maintainAspectRatio

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`my_FLVPlybk``.maintainAspectRatio`

Description

Propriété ; valeur booléenne qui, si elle est définie sur `true`, conserve les proportions de la vidéo. Si vous modifiez cette propriété de `false` à `true` et que la propriété `autoSize` est définie sur `false` une fois qu'un fichier FLV a été chargé, un redimensionnement automatique de la vidéo commence immédiatement. La valeur par défaut est `true`.

Exemple

L'exemple suivant appelle la méthode `setSize()` pour changer les dimensions de l'occurrence FLVPlayback, déclenchant un événement `resize`. La propriété `maintainAspectRatio`, dont la valeur par défaut est définie sur `true`, force un second événement `resize` pour conserver les proportions. Le gestionnaire d'événements resize affiche la largeur et la hauteur de l'occurrence FLVPlayback redimensionnée pour ces deux occurrences dans le panneau Sortie. Si vous définissez la propriété `maintainAspectRatio` sur `false`, les dimensions spécifiées par la méthode `setSize()` prennent effet.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
// Propriété maintainAspectRatio définie par défaut sur true,
// ce qui provoque le redimensionnement en cas de changement
// dans les dimensions.
// Suppression des séparateurs de commentaires sur la ligne suivante
// pour désactiver le redimensionnement.

// my_FLVPlybk.maintainAspectRatio = false;
var listenerObject:Object = new Object();
listenerObject.resize = function(eventObject:Object) {
  trace("resize event; Width is: " + eventObject.target.width + " Height
  is: " + eventObject.target.height);
};
my_FLVPlybk.addEventListener("resize", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
my_FLVPlybk.setSize(300, 300);
```

Voir aussi

`FLVPlayback.autoSize`, `FLVPlayback.height`, `FLVPlayback.preferredHeight`, `FLVPlayback.preferredWidth`, `FLVPlayback.resize`, `FLVPlayback.setSize()`, `FLVPlayback.width`

FLVPlayback.metadata

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.metadata

Description

Propriété : objet représentant un paquet d'informations de métadonnées reçu depuis un appel à la fonction de rappel NetSteam.onMetaData(), le cas échéant. Lecture seule.

Si le fichier FLV est codé à l'aide de l'encodeur de Flash CS3, la propriété metadata contient les informations présentées ci-dessous. Les fichiers FLV plus anciens contiennent uniquement les valeurs height, width et duration.

Paramètre	Description
canSeekToEnd	Valeur booléenne qui est définie sur true si le fichier FLV est codé avec une image-clé sur la dernière image qui permet de rechercher jusqu'à la fin d'un clip téléchargé progressivement. Elle est définie sur false si le fichier FLV n'est pas codé avec une image-clé sur la dernière image.
cuePoints	Tableau d'objets (un par point de repère intégré dans le fichier FLV). Cette valeur n'est pas définie si le fichier FLV ne contient pas de points de repère. Chaque objet possède les propriétés ci-dessous. • type Chaîne qui spécifie le type de point de repère : « navigation » ou « event ». • name Chaîne représentant le nom du point de repère. • time Nombre correspondant à l'heure du point de repère (en secondes) avec une précision de trois chiffres (millisecondes). • parameters Objet facultatif possédant des paires nom/valeur désignées par l'utilisateur au moment de la création des points de repère.
audiocodecid	Nombre qui indique le codec audio (technique de codage/décodage) utilisé.
audiodelay	Nombre qui indique quelle heure dans le fichier FLV « time 0 » du fichier FLV d'origine existe. Le contenu vidéo doit être légèrement retardé pour synchroniser correctement l'audio.
audiodatarate	Nombre indiquant les kilo-octets par seconde de l'audio.

Paramètre	Description
videocodecid	Nombre indiquant la version codec utilisée pour coder la vidéo.
framerate	Nombre indiquant la fréquence d'images du fichier FLV.
videodatarate	Nombre indiquant la vitesse de transmission vidéo du fichier FLV.
height	Nombre indiquant la hauteur du fichier FLV.
width	Nombre indiquant la largeur du fichier FLV.
duration	Nombre indiquant la durée du fichier FLV en secondes.

Exemple

L'exemple suivant affiche dans le panneau Sortie un échantillonnage des valeurs `metadata` du fichier FLV cuepoints.flv. Il affiche les données lorsque l'événement `metadataReceived` se produit.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.metadataReceived = function(eventObject:Object):Void {
    trace("canSeekToEnd is " + my_FLVPlybk.metadata.canSeekToEnd);
    trace("Number of cue points is " +
  my_FLVPlybk.metadata.cuePoints.length);
    trace("Frame rate is " + my_FLVPlybk.metadata.framerate);
    trace("Height is " + my_FLVPlybk.metadata.height);
    trace("Width is " + my_FLVPlybk.metadata.width);
    trace("Duration is " + my_FLVPlybk.metadata.duration + " seconds");
};
my_FLVPlybk.addEventListener("metadataReceived", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Voir aussi

FLVPlayback.metadataLoaded, FLVPlayback.metadataReceived

FLVPlayback.metadataLoaded

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.metadataLoaded

Description

Propriété : valeur booléenne définie sur `true` si un paquet de métadonnées a été trouvé et traité *ou* si le fichier FLV a été codé sans le paquet de métadonnées. En d'autres mots, la valeur est `true` si les métadonnées sont reçues ou si vous n'allez jamais en recevoir. De cette façon, vous savez si vous en avez et, si vous n'en avez pas, vous savez à quoi vous attendre. Si vous souhaitez simplement savoir si vous avez des métadonnées, vous pouvez vérifier la valeur :

```
FLVPlayback.metadata != null
```

Utilisez cette propriété pour vérifier si vous pouvez récupérer des informations utiles avec les méthodes pour trouver, puis activer ou désactiver des points de repère. Lecture seule.

Exemple

L'exemple suivant crée un écouteur pour l'événement `progress`. Lorsque l'événement se produit, l'exemple vérifie si la propriété `metadataLoaded` est définie sur `true`. Le cas échéant, il affiche les valeurs `height`, `width` et `duration` des métadonnées dans le panneau Sortie.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.progress = function(eventObject:Object):Void {
    if(my_FLVPlybk.metadataLoaded){
      trace("Height is " + my_FLVPlybk.metadata.height);
      trace("Width is " + my_FLVPlybk.metadata.width);
      trace("Duration is " + my_FLVPlybk.metadata.duration + " seconds");
    }
};
```

```
my_FLVPlybk.addEventListener("progress", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.metadata`, `FLVPlayback.metadataReceived`

FLVPlayback.metadataReceived

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.metadataReceived = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("metadataReceived", listenerObject);
```

Description

Evénement : distribué la première fois que les métadonnées du fichier FLV sont atteintes. L'objet événement possède une propriété `info` qui contient l'objet `info` reçu par le rappel `NetStream.onMetaData`.

L'objet événement a une propriété `vp`, qui correspond au numéro d'index du lecteur vidéo auquel cet événement s'applique. Pour plus d'informations, voir les sections « FLVPlayback.activeVideoPlayerIndex » à la page 571 et « FLVPlayback.visibleVideoPlayerIndex » à la page 713.

Exemple

L'exemple suivant crée un écouteur pour l'événement `metadataReceived`. Lorsque l'événement se produit, le gestionnaire d'événements envoie le nom, l'heure et le type de chaque point de repère décrit dans la propriété `metadata` vers le panneau Sortie.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.metadataReceived = function(eventObject:Object):Void {
     var i:Number = 0;
     trace("This FLV contains the following cue points:");
     while(i < my_FLVPlybk.metadata.cuePoints.length) {
       trace("\nName: " + my_FLVPlybk.metadata.cuePoints[i].name);
       trace(" Time: " + my_FLVPlybk.metadata.cuePoints[i].time);
       trace(" Type is " + my_FLVPlybk.metadata.cuePoints[i].type);
       ++i;
     }
};
my_FLVPlybk.addEventListener("metadataReceived", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Voir aussi

`FLVPlayback.metadata`, `FLVPlayback.metadataLoaded`

FLVPlayback.muteButton

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`my_FLVPlybk``.muteButton`

Description

Propriété : objet MovieClip qui correspond à la commande de bouton du son. Pour plus d'informations sur l'utilisation des composants de l'interface utilisateur personnalisée Lecture de fichiers FLV pour les commandes de lecture, reportez-vous à « Application d'enveloppe aux composants particuliers de l'interface utilisateur personnalisée Lecture de fichiers FLV », à la page 544.

Cliquer sur la commande muteButton distribue un événement `volumeUpdate`.

Exemple

L'exemple suivant utilise les propriétés `backButton`, `forwardButton`, `playPauseButton`, `stopButton` et `muteButton` pour associer des commandes d'interface utilisateur FLV personnalisées et individuelles à un composant FLVPlayback.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Dans l'inspecteur des composants, définissez le paramètre Skin (Enveloppe) sur None. Ajoutez les composants individuels suivants de l'interface utilisateur personnalisée FLV et donnez-leur les noms d'occurrence indiqués entre parenthèses : BackButton (**my_bkbttn**), ForwardButton (**my_fwdbttn**), PlayPauseButton (**my_plypausbttn**), StopButton (**my_stopbttn**) et MuteButton (**my_mutebttn**). Ajoutez ensuite les lignes de code suivantes dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
  - composants FLV Custom UI BackButton, ForwardButton, PlayPauseButton,
  StopButton et MuteButton dans la bibliothèque
*/
import mx.video.*;
my_FLVPlybk.backButton = my_bkbttn;
my_FLVPlybk.forwardButton = my_fwdbttn;
my_FLVPlybk.playPauseButton = my_plypausbttn;
my_FLVPlybk.stopButton = my_stopbttn;
my_FLVPlybk.muteButton = my_mutebttn;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.skin`, `FLVPlayback.volume`, `FLVPlayback.volumeBar`, `FLVPlayback.volumeUpdate`

FLVPlayback.NAVIGATION

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`mx.video.FLVPlayback.NAVIGATION`

Description

Propriété de la classe FLVPlayback en lecture seule qui contient la constante de type chaîne « `navigation` ». Vous pouvez utiliser cette propriété comme paramètre `type` des méthodes `findCuePoint()` et `findNearestCuePoint()`.

Exemple

L'exemple suivant utilise la propriété `FLVPlayback.NAVIGATION` pour indiquer le type de point de repère à rechercher.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
// Recherche d'un point de repère de navigation en utilisant l'heure.
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  var rtn_cuePt:Object = new Object();
  rtn_cuePt = my_FLVPlybk.findCuePoint(7.748, FLVPlayback.NAVIGATION);
  trace("Found cue point at " + rtn_cuePt.time + " of type " +
  rtn_cuePt.type);
}
my_FLVPlybk.addEventListener("ready", listenerObject)
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Voir aussi

`FLVPlayback.findCuePoint()`, `FLVPlayback.findNearestCuePoint()`

FLVPlayback.ncMgr

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`my_FLVPlybk``.ncMgr`

Description

Propriété : objet INCManager qui permet d'accéder à une occurrence de la classe implémentant `INCManager`, qui a une interface avec la classe `NCManager`.

Vous pouvez utiliser cette propriété pour implémenter un objet INCManager personnalisé qui requiert une initialisation personnalisée. Lecture seule.

Exemple

L'exemple suivant affiche la valeur de la propriété `DEFAULT_TIMEOUT` de NetConnection lorsque l'événement `ready` se produit.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
// Indication du nom et de l'emplacement du fichier FLV.
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  clouds.flv";
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  var NC:Object = new Object();
  NC = my_FLVPlybk.ncMgr;
  trace("Net connection timeout is " + NC.DEFAULT_TIMEOUT + "
  milliseconds");
};
my_FLVPlybk.addEventListener("ready", listenerObject);
```

Voir aussi

Classe VideoPlayer

FLVPlayback.pause()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVplybk.pause()

Paramètres

Aucun.

Renvoie

Aucune.

Description

Méthode : interrompt la lecture du flux vidéo en continu.

Exemple

L'exemple suivant crée un écouteur pour l'événement `playheadUpdate`. Lorsque cet événement se produit, le gestionnaire d'événements vérifie si le paramètre `playheadTime` est compris entre 5 et 5,05 secondes. Le cas échéant, le gestionnaire d'événements appelle la méthode `pause()` pour suspendre la lecture du fichier FLV. Le gestionnaire d'événements `paused` vous invite à appuyer sur le bouton Lire pour continuer.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Faites glisser un composant TextArea sur la scène, sous l'occurrence FLVPlayback, et nommez l'occurrence, TextArea **my_ta**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_ta.visible = false;
my_FLVPlybk.playheadUpdateInterval = 5;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.playheadUpdate = function(eventObject:Object):Void {
  if ((eventObject.playheadTime >= 5) && (eventObject.playheadTime < 5.05))
  {
    my_FLVPlybk.pause();
  }
};
my_FLVPlybk.addEventListener("playheadUpdate", listenerObject);
listenerObject.paused = function(eventObject:Object):Void {
  my_ta.text = "Paused; push Play to continue";
  my_ta.visible = true;
};
my_FLVPlybk.addEventListener("paused", listenerObject);
```

Voir aussi

`FLVPlayback.paused`, `FLVPlayback.play()`, `FLVPlayback.rewind`

FLVPlayback.pauseButton

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.pauseButton

Description

Propriété ; MovieClip étant la commande PauseButton. Pour plus d'informations sur l'utilisation des composants de l'interface utilisateur personnalisée Lecture de fichiers FLV pour les commandes de lecture, reportez-vous à « Application d'enveloppe aux composants particuliers de l'interface utilisateur personnalisée Lecture de fichiers FLV », à la page 544.

Exemple

L'exemple suivant utilise les propriétés `backButton`, `forwardButton`, `playButton`, `pauseButton` et `stopButton` pour associer des commandes individuelles de l'interface utilisateur personnalisée FLV à un composant FLVPlayback.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Dans l'inspecteur des composants, définissez le paramètre Skin (Enveloppe) sur None. Ajoutez les composants individuels suivants de l'interface utilisateur personnalisée FLV et donnez-leur les noms d'occurrence indiqués entre parenthèses : BackButton (**my_bkbttn**), ForwardButton (**my_fwdbttn**), PlayButton (**my_plybttn**), PauseButton (**my_pausbttn**) et StopButton (**my_stopbttn**). Ajoutez ensuite les lignes de code suivantes dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
  - Composants BackButton, ForwardButton, PlayButton, PauseButton et
   StopButton de l'interface utilisateur personnalisée FLV dans la
  bibliothèque
*/
import mx.video.*;
my_FLVPlybk.backButton = my_bkbttn;
my_FLVPlybk.forwardButton = my_fwdbttn;
my_FLVPlybk.playButton = my_plybttn;
my_FLVPlybk.pauseButton = my_pausbttn;
my_FLVPlybk.stopButton = my_stopbttn;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.playButton`, `FLVPlayback.playPauseButton`, `FLVPlayback.skin`

FLVPlayback.PAUSED

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
mx.video.FLVPlayback.PAUSED
```

Description

Propriété de la classe FLVPlayback en lecture seule qui contient la constante de type chaîne « `paused` ». Vous pouvez comparer cette propriété avec la propriété `state` pour déterminer si le composant est en état de pause.

Exemple

L'exemple suivant utilise la propriété `FLVPlayback.PAUSED` pour afficher l'état du fichier FLV lorsque l'utilisateur clique sur le bouton Pause.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  if(eventObject.state == FLVPlayback.PAUSED)
    trace("FLV is " + FLVPlayback.PAUSED);
}
my_FLVPlybk.addEventListener("stateChange", listenerObject)
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.paused

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.paused = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("paused", listenerObject);
```

Description

Evénement : distribué lorsque le lecteur entre en état de pause. Cela se produit lorsque vous appelez la méthode `pause()` ou cliquez sur la commande correspondante et également dans certains cas, lorsque le fichier FLV est chargé si le paramètre `autoPlay` est défini sur `false` (l'état risque de passer à `stopped`). L'objet événement possède les propriétés `state`, `playheadTime` et `vp`, qui correspond au numéro d'index du lecteur vidéo auquel cet événement s'applique. Pour plus d'informations sur la propriété `vp`, reportez-vous à « FLVPlayback.activeVideoPlayerIndex » à la page 571 et à « FLVPlayback.visibleVideoPlayerIndex » à la page 713.

L'événement `stateChange` est également distribué.

Exemple

L'exemple suivant crée un écouteur pour l'événement `playheadUpdate`. Lorsque cet événement a lieu, le gestionnaire d'événements vérifie si la propriété `playheadTime` est comprise entre 5 et 5,05 secondes. Le cas échéant, le gestionnaire d'événements appelle la méthode `pause()` pour suspendre la lecture du fichier FLV. Cela déclenche un événement `paused` pour lequel le gestionnaire d'événements `paused` affiche le message « The FLV is paused! ».

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.playheadUpdateInterval = 5;
var listenerObject:Object = new Object();
listenerObject.playheadUpdate = function(eventObject:Object):Void {
  if ((eventObject.playheadTime >= 5) && (eventObject.playheadTime < 5.05))
  {
    my_FLVPlybk.pause();
  }
}
my_FLVPlybk.addEventListener("playheadUpdate", listenerObject);
listenerObject.paused = function(eventObject:Object) {
  trace("FLV is " + my_FLVPlybk.state + "!");
};
my_FLVPlybk.addEventListener("paused", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.pause()`, `FLVPlayback.paused`, `FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.paused

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`my_FLVPlybk``.paused`

Description

Propriété : valeur booléenne définie sur `true` si le fichier FLV est en état de pause. Lecture seule.

Exemple

L'exemple suivant crée un écouteur pour l'événement `stateChange`. Lorsque l'événement a lieu, il vérifie la propriété `paused` pour déterminer si le composant est en état de pause. Le cas échéant, il affiche un message le signalant dans le panneau Sortie. Vous devez cliquer sur le bouton Pause lors de la lecture du fichier FLV pour provoquer l'état de pause.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object) {
  if(my_FLVPlybk.paused)
    trace("FLV is in " + FLVPlayback.PAUSED + " state");
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.paused`, `FLVPlayback.PAUSED`, `FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.play()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
my_FLVplybk.play ([contentPath:String, totalTime:Number, isLive:Boolean])
```

Paramètres

contentPath Chaîne indiquant l'URL du fichier FLV à diffuser en continu et de quelle façon. L'URL peut être un chemin local, l'URL HTTP d'un fichier FLV, l'URL RTMP d'un flux de fichier FLV ou l'URL HTTP d'un fichier XML. La propriété `contentPath` est facultative, mais elle doit être définie soit dans l'inspecteur de composants, soit via ActionScript. Autrement cette méthode n'a aucun effet.

totalTime Nombre représentant la durée de lecture totale de la vidéo. Facultatif.

isLive Valeur booléenne définie sur `true` si le flux vidéo est en direct. Cette valeur est significative uniquement lors de la diffusion en continu à partir d'un FMS ou FVSS. La valeur de cette propriété est ignorée s'il s'agit d'un téléchargement HTTP. Facultatif.

Renvoie

Aucune.

Description

Méthode ; lit le flux vidéo. Sans paramètre, la méthode fait simplement passer le fichier FLV d'un état de pause ou arrêté à l'état de lecture.

En cas d'utilisation de paramètres, la méthode agit comme un raccourci pour définir la propriété `autoPlay` sur `true` ainsi que les propriétés `isLive`, `totalTime` et `contentPath`. Si les propriétés `totalTime` et `isLive` ont la valeur undefined, elles ne sont pas définies.

Exemple

L'exemple suivant désactive la lecture automatique du fichier FLV, appelle la méthode `seekSeconds()` pour définir les 20 secondes de la tête de lecture dans la vidéo, puis la méthode `play()` pour démarrer la lecture du fichier FLV à ce stade.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.autoPlay = false;
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  my_FLVPlybk.seekSeconds(4);
  my_FLVPlybk.play();
};
my_FLVPlybk.addEventListener("ready", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.autoPlay`, `FLVPlayback.contentPath`, `FLVPlayback.load()`, `FLVPlayback.pause()`, `FLVPlayback.stop()`

FLVPlayback.playButton

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.playButton

Description

Propriété : objet MovieClip qui correspond au bouton Lire. Pour plus d'informations sur l'utilisation des composants de l'interface utilisateur personnalisée Lecture de fichiers FLV pour les commandes de lecture, reportez-vous à « Application d'enveloppe aux composants particuliers de l'interface utilisateur personnalisée Lecture de fichiers FLV », à la page 544.

Exemple

L'exemple suivant utilise les propriétés `backButton`, `forwardButton`, `playButton`, `pauseButton` et `stopButton` pour associer des commandes individuelles de l'interface utilisateur personnalisée FLV à un composant FLVPlayback.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez les composants individuels suivants de l'interface utilisateur personnalisée FLV et donnez-leur les noms d'occurrence indiqués entre parenthèses : BackButton (**my_bkbttn**), ForwardButton (**my_fwdbttn**), PlayButton (**my_plybttn**), PauseButton (**my_pausbttn**) et StopButton (**my_stopbttn**). Ajoutez ensuite les lignes de code suivantes au panneau Actions :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
  - Composants BackButton, ForwardButton, PlayButton, PauseButton et
   StopButton de l'interface utilisateur personnalisée FLV dans la
  bibliothèque
*/
import mx.video.*;
my_FLVPlybk.backButton = my_bkbttn;
my_FLVPlybk.forwardButton = my_fwdbttn;
my_FLVPlybk.playButton = my_plybttn;
my_FLVPlybk.pauseButton = my_pausbttn;
my_FLVPlybk.stopButton = my_stopbttn;
```

Voir aussi

`FLVPlayback.playing`, `FLVPlayback.skin`

FLVPlayback.playheadPercentage

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.playheadPercentage

Description

Propriété : nombre qui indique le paramètre `playheadTime` actuel sous la forme d'un pourcentage de la propriété `totalTime`. Si vous accédez à cette propriété, elle contient le pourcentage de durée de lecture écoulée. Si vous la définissez, cela provoque une recherche jusqu'au point représentant le pourcentage de durée de lecture du fichier FLV.

La valeur de cette propriété est proportionnelle à celle de la propriété `totalTime`.

Le composant génère une erreur VideoError si vous indiquez un pourcentage non valide ou si la propriété `totalTime` n'est pas définie, a une valeur nulle ou négative.

Exemple

L'exemple suivant affiche le pourcentage du fichier FLV qui a été lu au moment où le point de repère `point2` se présente. Au niveau du point de repère `point3`, il définit le paramètre `playheadPercentage` sur `10`, ce qui provoque une opération de recherche jusqu'au point représentant 10 % de la lecture depuis le début du fichier FLV et crée une boucle de lecture.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object):Void {
    if(eventObject.info.name == "point2")
      trace("point2 occurred at " + my_FLVPlybk.playheadPercentage + "
  percent of FLV");
    if(eventObject.info.name == "point3")
      my_FLVPlybk.playheadPercentage = 10;
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
```

Voir aussi

`FLVPlayback.playheadTime`, `FLVPlayback.seekPercent()`, `FLVPlayback.totalTime`

FLVPlayback.playheadTime

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.playheadTime

Description

Propriété : nombre représentant la durée de lecture (ou position de la tête de lecture) actuelle, mesurée en secondes. Il peut s'agir d'une valeur décimale. Lorsque vous définissez cette propriété, vous déclenchez une recherche et vous avez toutes les restrictions d'une recherche.

Lorsque la position de la tête de lecture change (une fois toutes les 0,25 secondes lors de la lecture du fichier FLV), le composant distribue l'événement `playheadUpdate`.

La propriété `playheadTime` peut ne pas avoir la valeur attendue immédiatement après l'appel aux méthodes de recherche ou la définition de la propriété `playheadTime` pour provoquer une recherche. Il y a là plusieurs raisons. Tout d'abord, s'il s'agit d'un téléchargement progressif, vous pouvez rechercher uniquement une image-clé. Ainsi une recherche vous amène à l'heure de la première image-clé après l'heure spécifiée. (En cas de diffusion à flux continu, une recherche vous conduit toujours à l'heure exacte spécifiée même si le fichier FLV source n'y possède pas d'image-clé.) En second lieu, une recherche est asynchrone. Ainsi si vous appelez une méthode de recherche ou définissez la propriété `playheadTime`, celle-ci n'est pas immédiatement mise à jour. Pour obtenir l'heure à la fin de la recherche, écoutez l'événement `seek`, qui ne se déclenche pas tant que la propriété `playheadTime` n'a pas été mise à jour.

Exemple

L'exemple suivant utilise des occurrences de l'événement `stateChange` lorsqu'il se produit pendant la lecture du fichier FLV, et affiche la durée de lecture écoulée dans le panneau Sortie.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
```

```
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  trace(my_FLVPlybk.state + ": playhead time is: " +
  my_FLVPlybk.playheadTime);
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.playheadUpdate`, `FLVPlayback.playheadUpdateInterval`, `FLVPlayback.seek()`, `FLVPlayback.stateChange`

FLVPlayback.playheadUpdate

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.playheadUpdate = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("playheadUpdate", listenerObject);
```

Description

Evénement : distribué alors que le fichier FLV est en cours de lecture à la fréquence spécifiée par la propriété `playheadUpdateInterval`. La valeur par défaut est 0,25 secondes.
Le composant ne distribue pas cet événement lorsque le lecteur vidéo est en pause ou arrêté, à moins qu'une recherche n'ait lieu. L'objet événement possède les propriétés `state`, `playheadTime` et `vp`.

Exemple

L'exemple suivant utilise des occurrences de l'événement `playheadUpdate` lorsqu'il se produit pendant la lecture du fichier FLV, et affiche la durée de lecture écoulée dans le panneau Sortie.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.playheadUpdate = function(eventObject:Object):Void {
  trace(my_FLVPlybk.state + ": playhead time is: " +
  eventObject.playheadTime);
};
my_FLVPlybk.addEventListener("playheadUpdate", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.playheadTime`, `FLVPlayback.playheadUpdateInterval`

FLVPlayback.playheadUpdateInterval

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`my_FLVPlybk``.playheadUpdateInterval`

Description

Propriété : nombre représentant la durée (en millisecondes) entre chaque événement `playheadUpdate`. Lorsque vous définissez cette propriété pendant la lecture du fichier FLV, l'horloge redémarre. La valeur par défaut est 250.

Etant donné que les points de repère ActionScript démarrent aux mises à jour de la tête de lecture, diminuer la valeur de la propriété `playheadUpdateInterval` peut améliorer la précision des points de repère ActionScript.

Comme l'intervalle des mises à jour de la tête de lecture est défini par un appel à la fonction globale `setInterval()`, la mise à jour ne peut pas être déclenchée plus fréquemment que la fréquence d'images du fichier FLV, comme pour tout intervalle défini de cette façon.
Par exemple, pour la fréquence par défaut de 12 images par seconde, l'intervalle efficace le plus petit que vous pouvez créer est approximativement de 83 millisecondes ou d'une seconde (1 000 millisecondes) divisée par 12.

Exemple

L'exemple suivant définit la propriété `playheadUpdateInterval` sur 3 000 et crée un écouteur qui utilise des occurrences de l'événement `playheadUpdate` lorsqu'il a lieu pendant la lecture du fichier FLV. Lorsque l'événement a lieu, le gestionnaire d'événements affiche la durée de lecture écoulée dans le panneau Sortie.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.playheadUpdateInterval = 3000;
var listenerObject:Object = new Object();
listenerObject.playheadUpdate = function(eventObject:Object):Void {
  trace("playhead time is: " + eventObject.playheadTime);
};
my_FLVPlybk.addEventListener("playheadUpdate", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.playheadTime`, `FLVPlayback.playheadUpdate`

FLVPlayback.PLAYING

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
mx.video.FLVPlayback.PLAYING
```

Description

Propriété de la classe FLVPlayback en lecture seule qui contient la constante de type chaîne « `playing` ». Vous pouvez comparer cette propriété avec la propriété `state` pour déterminer si le composant est en état de lecture.

Exemple

L'exemple suivant utilise la propriété `FLVPlayback.PLAYING` pour voir si l'état est bien « `playing` » lorsqu'un événement `stateChange` se produit. Il contient également la constante comme partie d'un message dans le panneau Sortie.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
    if(eventObject.state == FLVPlayback.PLAYING)
      trace(my_FLVPlybk.contentPath + " is now " + FLVPlayback.PLAYING);
}
my_FLVPlybk.addEventListener("stateChange", listenerObject);
```

Voir aussi

`FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.playing

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var :Object = new Object();
listenerObject.playing = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("playing", listenerObject);
```

Description

Evénement : distribué lorsque le composant entre dans l'état de lecture. Cela risque de ne pas se produire immédiatement après que vous avez appelé la méthode `play()` ou cliqué sur la commande correspondante. Souvent, le composant entre d'abord dans l'état de mise en mémoire tampon, puis dans l'état de lecture. L'objet événement possède les propriétés `state`, `playheadTime` et `vp`, qui correspond au numéro d'index du lecteur vidéo auquel cet événement s'applique. Pour plus d'informations sur la propriété `vp`, reportez-vous à « FLVPlayback.activeVideoPlayerIndex » à la page 571 et à « FLVPlayback.visibleVideoPlayerIndex » à la page 713.

L'occurrence du composant FLVPlayback distribue également l'événement `stateChange`.

Exemple

L'exemple suivant affiche la valeur de la propriété `contentPath` dans une zone de texte lorsque l'événement `playing` se produit.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Faites glisser un composant TextArea sur la scène, sous l'occurrence FLVPlayback, et nommez l'occurrence, TextArea **my_ta**. Ajoutez ensuite le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.playing = function(eventObject:Object):Void {
    my_ta.text = "Now playing: " + my_FLVPlybk.contentPath;
}
my_FLVPlybk.addEventListener("playing", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.play()`,`FLVPlayback.playing`, `FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.playing

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.playing

Description

Propriété : valeur booléenne définie sur `true` si le fichier FLV est en état de lecture. Lecture seule.

Exemple

L'exemple suivant écoute les occurrences de l'événement `stateChange` lorsqu'il a lieu pendant la lecture du fichier FLV. Lorsque l'événement a lieu, l'exemple affiche la valeur de la propriété `playing` dans le panneau Sortie.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
trace(my_FLVPlybk.state + ": playing property is " + my_FLVPlybk.playing);
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  trace(my_FLVPlybk.state + ": playing property is " +
  my_FLVPlybk.playing);
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.playing`, `FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.playPauseButton

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.playPauseButton

Description

Propriété : objet MovieClip qui correspond à la commande PlayPauseButton. Pour plus d'informations sur l'utilisation des composants de l'interface utilisateur personnalisée Lecture de fichiers FLV pour les commandes de lecture, reportez-vous à « Application d'enveloppe aux composants particuliers de l'interface utilisateur personnalisée Lecture de fichiers FLV », à la page 544.

Exemple

L'exemple suivant utilise les propriétés `playPauseButton`, `stopButton`, `backButton` et `forwardButton` pour associer des commandes individuelles de l'interface utilisateur personnalisée FLV à un composant FLVPlayback.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez les composants individuels suivants de l'interface utilisateur personnalisée FLV et donnez-leur les noms d'occurrence indiqués entre parenthèses : BackButton (**my_bkbttn**), ForwardButton (**my_fwdbttn**), PlayPauseButton (**my_plypausebttn**) et StopButton (**my_stopbttn**). Ajoutez ensuite les lignes de code suivantes au panneau Actions :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
  - composants FLV Custom UI PlayPauseButton, StopButton, BackButton et
  ForwardButton dans la bibliothèque
*/
import mx.video.*;
my_FLVPlybk.playPauseButton = my_plypausbttn;
my_FLVPlybk.stopButton = my_stopbttn;
my_FLVPlybk.backButton = my_bkbttn;
my_FLVPlybk.forwardButton = my_fwdbttn;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.playButton`, `FLVPlayback.playPauseButton`, `FLVPlayback.paused`, `FLVPlayback.skin`

FLVPlayback.preferredHeight

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.preferredHeight

Description

Propriété : nombre indiquant la hauteur du fichier FLV source. Ces informations ne sont pas valides dès que vous appelez les méthodes `play()` ou `load()`. Elles le sont lorsque l'événement `ready` est déclenché. Si la valeur des propriétés `autoSize` ou `maintainAspectRatio` est définie sur `true`, il est préférable de la lire lorsque l'événement `resize` est déclenché. Lecture seule.

Exemple

L'exemple suivant définit la dimension de l'occurrence FLVPlayback lorsque l'événement `ready` se produit. Lorsque l'événement `cuePoint` se produit, il redéfinit la dimension sur celle spécifiée par les propriétés `preferredHeight` et `preferredWidth`.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.resize = function(eventObject:Object):Void {
    trace("width is: " + my_FLVPlybk.width);
    trace("height is: " + my_FLVPlybk.height);
};
my_FLVPlybk.addEventListener("resize", listenerObject);
listenerObject.ready = function(eventObject:Object):Void {
    my_FLVPlybk.setSize(250, 350);
};
```

```
my_FLVPlybk.addEventListener("ready", listenerObject);
listenerObject.cuePoint = function(eventObject:Object):Void {
    my_FLVPlybk.setSize(my_FLVPlybk.preferredWidth,
  my_FLVPlybk.preferredHeight);
};
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
my_FLVPlybk.addASCuePoint(1.5, "AScp1");
```

Voir aussi

`FLVPlayback.autoSize`, `FLVPlayback.height`, `FLVPlayback.maintainAspectRatio`, `FLVPlayback.preferredWidth`, `FLVPlayback.ready`, `FLVPlayback.setSize()`, `FLVPlayback.setScale()`, `FLVPlayback.width`

FLVPlayback.preferredWidth

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`my_FLVPlybk``.preferredWidth`

Description

Propriété : indique la largeur du fichier FLV source. Ces informations ne sont pas valides dès que vous appelez les méthodes `play()` ou `load()` ; elles le sont lorsque l'événement `ready` est déclenché. Si la valeur des propriétés `autoSize` ou `maintainAspectRatio` est définie sur `true`, il est préférable de la lire lorsque l'événement `resize` est déclenché. Lecture seule.

Exemple

L'exemple suivant définit la dimension de l'occurrence FLVPlayback lorsque l'événement `ready` se produit. Lorsque l'événement `cuePoint` se produit, il redéfinit la dimension sur celle spécifiée par les propriétés `preferredHeight` et `preferredWidth`.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.resize = function(eventObject:Object):Void {
    trace("width is: " + my_FLVPlybk.width);
    trace("height is: " + my_FLVPlybk.height);
};
my_FLVPlybk.addEventListener("resize", listenerObject);
listenerObject.ready = function(eventObject:Object):Void {
    my_FLVPlybk.setSize(250, 350);
};
my_FLVPlybk.addEventListener("ready", listenerObject);
listenerObject.cuePoint = function(eventObject:Object):Void {
    my_FLVPlybk.setSize(my_FLVPlybk.preferredWidth,
  my_FLVPlybk.preferredHeight);
};
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
my_FLVPlybk.addASCuePoint(1.5, "AScp1");
```

Voir aussi

FLVPlayback.autoSize, FLVPlayback.height, FLVPlayback.maintainAspectRatio, FLVPlayback.preferredHeight, FLVPlayback.ready, FLVPlayback.setSize(), FLVPlayback.setScale(), FLVPlayback.width

FLVPlayback.progress

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.progress = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("progress", listenerObject);
```

Description

Evénement : distribué à la fréquence spécifiée par la propriété `progressInterval`, il se déclenche au début du chargement et se termine lorsque tous les octets sont chargés ou en cas d'erreur réseau. La valeur par défaut est 0,25 secondes.

Il est distribué uniquement s'il s'agit d'un téléchargement HTTP progressif. Il indique la progression en nombre d'octets téléchargés. L'objet événement possède les propriétés `bytesLoaded` et `bytesTotal`, qui sont identiques aux propriétés FLVPlayback du même nom.

L'objet événement possède également la propriété `vp`, qui correspond au numéro d'index du lecteur vidéo auquel cet événement s'applique. Pour plus d'informations sur la propriété `vp`, reportez-vous à `FLVPlayback.activeVideoPlayerIndex` et à `FLVPlayback.visibleVideoPlayerIndex`.

Exemple

L'exemple suivant définit la propriété `progressInterval` sur 001 milliseconde, car le fichier FLV est petit, puis affiche le nombre d'octets chargés pour chaque occurrence de l'événement `progress`.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.progressInterval = 001;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.progress = function(eventObject:Object):Void {
    trace(eventObject.bytesLoaded);
}
my_FLVPlybk.addEventListener("progress", listenerObject);
```

Voir aussi

`FLVPlayback.activeVideoPlayerIndex`, `FLVPlayback.addEventListener()`, `FLVPlayback.bytesLoaded`, `FLVPlayback.bytesTotal`, `FLVPlayback.progressInterval`, `FLVPlayback.visibleVideoPlayerIndex`

FLVPlayback.progressInterval

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.progressInterval

Description

Propriété : nombre représentant la durée (en millisecondes) entre chaque événement `progress`. Si vous définissez cette propriété pendant la lecture du flux vidéo, l'horloge redémarre. La valeur par défaut est 250.

Exemple

L'exemple suivant définit la propriété `progressInterval` sur 001 milliseconde, car le fichier FLV est petit, puis affiche le nombre d'octets chargés pour chaque occurrence de l'événement `progress`.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.progressInterval = 001;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.progress = function(eventObject:Object):Void {
    trace(eventObject.bytesLoaded);
}
my_FLVPlybk.addEventListener("progress", listenerObject);
```

Voir aussi

FLVPlayback.progress

FLVPlayback.ready

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("ready", listenerObject);
```

Description

Evénement : distribué lorsque le fichier FLV est chargé et prêt à s'afficher. Il est déclenché la première fois que vous entrez un état réactif après avoir chargé un nouveau fichier FLV avec la méthode `play()` ou `load()`. Il est déclenché une seule fois par fichier FLV chargé.

L'objet événement possède les propriétés `state`, `playheadTime` et `vp`. La propriété `vp` correspond au numéro d'index du lecteur vidéo auquel cet événement s'applique. Pour plus d'informations sur la propriété `vp`, reportez-vous à « FLVPlayback.activeVideoPlayerIndex » à la page 571 et à « FLVPlayback.visibleVideoPlayerIndex » à la page 713.

Exemple

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
  - Composant TextArea sur la scène dont l'occurrence est nommée my_ta
*/
import mx.video.*;
my_ta.visible = false;
my_FLVPlybk.autoPlay = false;
my_ta.setSize(260, 30);
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
    my_ta.text = "The FLV is ready. Push Play to start playing";
    my_ta.visible = true;
};
my_FLVPlybk.addEventListener("ready", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.activeVideoPlayerIndex`, `FLVPlayback.addEventListener()`, `FLVPlayback.state`, `FLVPlayback.visibleVideoPlayerIndex`

FLVPlayback.removeASCuePoint()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
my_FLVplybk.removeASCuePoint(CuePoint:Object):Object
my_FLVplybk.removeASCuePoint(time:Number):Object
my_FLVplybk.removeASCuePoint(name:String):Object
```

Paramètres

cuePoint Objet point de repère possédant les propriétés `time` et `name` correspondant au point de repère à supprimer. Cette méthode n'active aucune autre propriété sur l'objet point de repère entrant. Si le paramètre `time` ou `name` a la valeur `null` ou n'est pas défini, la méthode utilise uniquement la propriété disponible. Si seul le paramètre `name` est fourni, la méthode supprime le premier point de repère portant ce nom.

time Nombre contenant l'heure du point de repère à supprimer. Cette méthode supprime le premier point de repère indiquant cette heure.

name Chaîne contenant le nom du point de repère à supprimer. Cette méthode supprime le premier point de repère portant ce nom.

Renvoie

Objet point de repère qui a été supprimé. Si aucun point de repère correspondant n'a été trouvé, la méthode renvoie la valeur `null`.

Description

Méthode : supprime un point de repère ActionScript du fichier FLV actuellement chargé. Seules les propriétés `name` et `time` sont utilisées dans le paramètre *CuePoint* pour rechercher le point de repère à supprimer.

Si plusieurs points de repère ActionScript correspondent aux critères de recherche, un seul est supprimé. Pour les supprimer tous, appelez plusieurs fois cette fonction dans une boucle avec les mêmes paramètres jusqu'à ce qu'elle renvoie la valeur null.

Les informations sur les points de repère sont effacées lorsque la propriété `contentPath` est définie. Par conséquent, pour définir de telles informations pour le fichier FLV suivant à charger, définissez d'abord `contentPath`.

Exemple

L'exemple suivant ajoute un point de repère ActionScript au fichier FLV, puis appelle la méthode `removeASCuePoint()` pour le supprimer. Il affiche le nom du point de repère supprimé dans le panneau Sortie.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
// Création d'un objet point de repère.
var cuePt:Object = new Object(); // Création d'un objet point de repère.
var rtn_cuePt:Object = new Object(); // Création d'un objet pour la
                                     // valeur renvoyée.
cuePt.time = 4.444;
cuePt.name = "ripples";
my_FLVPlybk.addASCuePoint(cuePt);  // Ajout d'un point de repère AS.
if ((rtn_cuePt = my_FLVPlybk.removeASCuePoint(cuePt)) != null) {
  trace("Removed cue point: " + rtn_cuePt.name);
}
```

Voir aussi

`FLVPlayback.addASCuePoint()`, `FLVPlayback.findCuePoint()`, `FLVPlayback.findNearestCuePoint()`, `FLVPlayback.findNextCuePointWithName()`

FLVPlayback.removeEventListener()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
my_FLVPlybk.removeEventListener(event:String, listener:Object):Void
my_FLVPlybk.removeEventListener(event:String, listener:Function):Void
```

Paramètres

event Chaîne qui indique le nom de l'événement pour lequel vous supprimez un écouteur.

listener Référence à l'objet écouteur ou à la fonction d'écoute que vous supprimez.

Renvoie

Aucune.

Description

Méthode : supprime un écouteur d'événement dans une occurrence de composant.

Exemple

L'exemple suivant supprime l'écouteur d'événement `cuePoint` lorsque le premier point de repère se présente. Seul le premier des trois points de repère peut donc être détecté.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
Usage 1: listener object
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
var listenerObject:Object = new Object(); // Création d'un objet écouteur.
listenerObject.cuePoint = function(eventObject:Object):Void {
    trace("Hit cue point at " + eventObject.info.time);
    my_FLVPlybk.removeEventListener("cuePoint", listenerObject);
};
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
```

```
Usage 2: listener function
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_ta.visible = false;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
function cuePoint(eventObject:Object):Void {
    trace("Hit cue point at " + eventObject.info.time);
    my_FLVPlybk.removeEventListener("cuePoint", cuePoint);
};
my_FLVPlybk.addEventListener("cuePoint", cuePoint);
```

Voir aussi

`FLVPlayback.addEventListener()`

FLVPlayback.resize

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.resize = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("resize", listenerObject);
```

Description

Evénement ; distribué lors du redimensionnement de la vidéo. Il se produit lorsque vous définissez la propriété `visibleVideoPlayerIndex`, puis basculez vers un lecteur vidéo dont les dimensions sont différentes. L'objet événement possède les propriétés `auto`, `x`, `y`, `width`, `height` et `vp`, qui correspond au numéro d'index du lecteur vidéo auquel cet événement s'applique. Pour plus d'informations sur la propriété `vp`, reportez-vous à « FLVPlayback.activeVideoPlayerIndex » à la page 571 et à « FLVPlayback.visibleVideoPlayerIndex » à la page 713.

La propriété `auto` est définie sur `true` lorsque le redimensionnement est automatique, car la propriété `autoSize` ou `maintainAspectRatio` est définie sur `true`. Dans ce cas, l'événement risque d'être distribué pour un lecteur vidéo autre celui qui est affiché. Il risque d'être distribué même si les dimensions ne changent pas réellement après une tentative de redimensionnement automatique du composant.

Lorsque la propriété `auto` est définie sur `false`, l'événement s'applique toujours au lecteur vidéo visible. La propriété `vp` apparaît encore, mais est toujours égale à la propriété `visibleVideoPlayerIndex`.

Le composant distribue l'événement (la propriété `auto` étant définie sur `false`) lorsque vous définissez la propriété `visibleVideoPlayerIndex` si vous basculez vers un lecteur vidéo dont les dimensions sont différentes de celles du lecteur actuellement visible.

Exemple

L'exemple suivant lit deux fichiers FLV. Il ajoute un point de repère ActionScript au premier fichier FLV, et lorsque l'événement `cuePoint` se produit, il bascule vers un second lecteur vidéo plus petit pour lire le second fichier FLV. Lorsqu'il définit la propriété `visibleVideoPlayerIndex` pour le lecteur vidéo, il déclenche l'événement `resize` qui affiche la taille et l'emplacement du lecteur vidéo actuel.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
// Désactivation des propriétés autoSize et maintainAspectRatio.
my_FLVPlybk.autoSize = false;
my_FLVPlybk.maintainAspectRatio = false;
// Lecture de ce fichier FLV.
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  clouds.flv";
// Ajout d'un point de repère.
my_FLVPlybk.addASCuePoint(3, "switch_here");
var listenerObject:Object = new Object();// Création d'un écouteur.
listenerObject.cuePoint = function(eventObject:Object):Void {
  // Ajout d'un second lecteur vidéo.
  my_FLVPlybk.activeVideoPlayerIndex = 1;
  // Lecture de ce fichier FLV.
  my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
  // Modification de la taille de ce lecteur vidéo.
  my_FLVPlybk.setSize(240, 180);
  my_FLVPlybk.visibleVideoPlayerIndex = 1; // Activation de sa visibilité.
  my_FLVPlybk.play(); // Lecture du fichier FLV.
};
// Ajout d'un écouteur pour un événement cuePoint.
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
listenerObject.resize = function(eventObject:Object):Void {
  // Affichage de l'emplacement et des dimensions.
  trace("Video player is #" + my_FLVPlybk.activeVideoPlayerIndex);
  trace("X coordinate is: " + eventObject.x);
  trace("Y coordinate is: " + eventObject.y);
  trace("Width is: " + eventObject.width);
  trace("Height is: " + eventObject.height);
};
// Ajout d'un écouteur pour un événement resize.
my_FLVPlybk.addEventListener("resize", listenerObject);
```

Voir aussi

FLVPlayback.activeVideoPlayerIndex, FLVPlayback.autoSize, FLVPlayback.height, FLVPlayback.maintainAspectRatio, FLVPlayback.preferredHeight, FLVPlayback.preferredWidth, FLVPlayback.setSize(), FLVPlayback.state, FLVPlayback.width, FLVPlayback.x, FLVPlayback.y

FLVPlayback.rewind

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.rewind = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("rewind", listenerObject);
```

Description

Evénement : distribué lorsque l'emplacement de la tête de lecture recule suite à un appel à la méthode `seek()` ou à la fin du rembobinage automatique.

L'objet événement possède les propriétés `auto`, `state` et `playheadTime`. Si l'événement résulte d'une recherche en arrière, la propriété `auto` est définie sur `false`. S'il résulte d'un rembobinage automatique, la propriété `auto` est définie sur `true`.

La propriété `playheadTime` reflète l'heure de destination.

L'événement `stateChange` est distribué avec l'état « `rewinding` » en cas de rembobinage automatique. Il ne se produit pas avant la fin du rembobinage. L'événement `seek` est distribué si le rembobinage a lieu pendant la recherche. L'occurrence FLVPlayback distribue également l'événement `playheadUpdate` lors du rembobinage.

L'événement `rewind` possède la propriété `vp`, qui correspond au numéro d'index du lecteur vidéo auquel cet événement s'applique. Pour plus d'informations sur la propriété `vp`, reportez-vous à `FLVPlayback.activeVideoPlayerIndex` et à `FLVPlayback.visibleVideoPlayerIndex`.

Exemple

L'exemple suivant définit la propriété `autoRewind` sur `true` et écoute l'événement `rewind`. Si cet événement se produit, le gestionnaire d'événements affiche les valeurs des propriétés `vp`, `state` et `playheadTime` dans le panneau Sortie.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.autoRewind = true;
var listenerObject:Object = new Object();
listenerObject.rewind = function(eventObject:Object) {
  trace("Video player is #" + eventObject.vp);
  trace("State is: " + eventObject.state);
  trace("Playhead time is: " + eventObject.playheadTime);
};
my_FLVPlybk.addEventListener("rewind", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.activeVideoPlayerIndex`, `FLVPlayback.playheadTime`, `FLVPlayback.playheadUpdateFLVPlayback.seek()`, `FLVPlayback.seekPercent()`, `FLVPlayback.seekSeconds()`, `FLVPlayback.seekToNavCuePoint()`, `FLVPlayback.seekToPrevNavCuePoint()`, `FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.REWINDING

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
mx.video.FLVPlayback.REWINDING
```

Description

Propriété de la classe FLVPlayback en lecture seule qui contient la constante de type chaîne « `rewinding` ». Vous pouvez comparer cette propriété avec la propriété `state` pour déterminer si le composant est en état de rembobinage.

Exemple

L'exemple suivant crée un écouteur d'événement `stateChange` et utilise la propriété `FLVPlayback.REWINDING` pour déterminer si le composant est en état de rembobinage.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;

var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  if(eventObject.state == FLVPlayback.REWINDING)
    trace("The current state is " + FLVPlayback.REWINDING);
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.scaleX

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`my_FLVPlybk``.scaleX`

Description

Propriété ; chiffre indiquant le redimensionnement horizontal. Le redimensionnement standard est 100.

Exemple

L'exemple suivant définit les propriétés (`scaleX`) (axe horizontal) et (`scaleY`) (axe vertical) de l'occurrence FLVPlayback sur 150 pour cent.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.scaleX = 150;
my_FLVPlybk.scaleY = 150;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.setScale()`, `FLVPlayback.scaleY`

FLVPlayback.scaleY

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`my_FLVPlybk``.scaleY`

Description

Propriété ; chiffre indiquant le redimensionnement vertical. Le redimensionnement standard est 100.

Exemple

L'exemple suivant définit le redimensionnement horizontal (`scaleX`) et vertical (`scaleY`) de l'occurrence FLVPlayback sur 150 pour cent.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.scaleX = 150;
my_FLVPlybk.scaleY = 150;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.scaleX`, `FLVPlayback.setScale()`

FLVPlayback.scrubbing

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`my_FLVPlybk``.scrubbing`

Description

Propriété : valeur booléenne qui est définie sur `true` si l'utilisateur effectue une modulation avec la commande SeekBar. Dans le cas contraire, elle est définie sur `false`. Lecture seule.

La *modulation* désigne le fait de saisir la poignée de la barre de recherche et de la faire glisser dans l'une ou l'autre direction afin de rechercher une séquence particulière du fichier FLV.

Exemple

L'exemple suivant affiche la valeur de la propriété `scrubbing` lorsqu'un événement `seek` se produit.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

REMARQUE	Vous devez saisir la poignée de la barre de recherche, la faire glisser, puis la relâcher pour provoquer l'événement.

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.seek = function(eventObject:Object):Void {
  if(my_FLVPlybk.scrubbing)
    trace("User is scrubbing at: " + eventObject.playheadTime);
};
my_FLVPlybk.addEventListener("seek", listenerObject);
```

Voir aussi

`FLVPlayback.seek`, `FLVPlayback.seekBar`, `FLVPlayback.scrubFinish`, `FLVPlayback.scrubStart`

FLVPlayback.scrubFinish

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.scrubFinish = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("scrubFinish", listenerObject);
```

Description

Evénement : distribué lorsque l'utilisateur arrête la modulation du fichier FLV avec la barre de recherche. La modulation désigne le fait de saisir la poignée de la barre de recherche et de la faire glisser dans l'une ou l'autre direction afin de rechercher une séquence particulière du fichier FLV. Elle s'arrête lorsque l'utilisateur relâche la poignée de la barre de recherche.

L'objet événement a les propriétés `state` et `playheadTime`. L'état passe en « `seeking` » à la fin de la modulation.

Le composant distribue également l'événement `stateChange` dont la propriété state équivaut au nouvel état. Il peut s'agir des états « `playing` », « `paused` », « `stopped` » ou « `buffering` ».

Exemple

L'exemple suivant écoute l'événement `scrubFinish` et affiche l'heure de fin de la modulation.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

REMARQUE	Vous devez saisir la poignée de la barre de recherche, la faire glisser, puis la relâcher pour provoquer l'événement.

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.scrubFinish = function(eventObject:Object):Void {
  trace("Scrubbing stopped at " + eventObject.playheadTime);
  trace("Current state is " + eventObject.state);
};
my_FLVPlybk.addEventListener("scrubFinish", listenerObject);
```

Voir aussi

`FLVPlayback.playheadTime`, `FLVPlayback.seek`, `FLVPlayback.seekBar`, `FLVPlayback.scrubStart`, `FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.scrubStart

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.scrubStart = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("scrubStart", listenerObject);
```

Description

Evénement : distribué lorsque l'utilisateur commence la modulation du fichier FLV avec la barre de recherche. La modulation désigne le fait de saisir la poignée de la barre de recherche et de la faire glisser dans l'une ou l'autre direction afin de rechercher une séquence particulière du fichier FLV. La modulation commence lorsque l'utilisateur clique sur la poignée de la barre de recherche et se termine lorsqu'il la relâche.

L'objet événement a les propriétés `state` et `playheadTime`.

Le composant distribue également l'événement `stateChange`, la propriété `state` étant définie sur « `seeking` ». Cette propriété reste ainsi définie jusqu'à ce que l'utilisateur arrête la modulation.

Exemple

L'exemple suivant écoute l'événement `scrubStart` et affiche l'heure de début de la modulation.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

REMARQUE	Vous devez saisir la poignée de la barre de recherche, puis la faire glisser pour provoquer l'événement.

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.scrubStart = function(eventObject:Object):Void {
  trace("Scrubbing began at " + eventObject.playheadTime);
};
my_FLVPlybk.addEventListener("scrubStart", listenerObject);
```

Voir aussi

FLVPlayback.playheadTime, FLVPlayback.scrubbing, FLVPlayback.scrubFinish, FLVPlayback.seekBar, FLVPlayback.state, FLVPlayback.stateChange

FLVPlayback.seek

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.seek = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("seek", listenerObject);
```

Description

Evénement : distribué lorsque l'emplacement de la tête de lecture est modifié suite à un appel à la méthode `seek()`, à la définition de la propriété `playheadTime` ou à l'utilisation de la commande seekBar. La propriété `playheadTime` reflète l'heure de destination. L'objet événement possède les propriétés `state`, `playheadTime` et `vp`, qui correspond au numéro d'index du lecteur vidéo auquel cet événement s'applique.

L'occurrence FLVPlayback distribue l'événement `rewind` lorsque la recherche s'effectue vers l'arrière et l'événement `fastForward` dans le cas contraire. Il distribue également l'événement `playheadUpdate`.

La propriété `playheadTime` peut ne pas avoir la valeur attendue immédiatement après l'appel aux méthodes de recherche ou après la définition de la propriété `playheadTime` pour provoquer une recherche. Tout d'abord, s'il s'agit d'un téléchargement progressif, vous pouvez rechercher uniquement une image-clé. Ainsi, une recherche vous amène à l'heure de la première image-clé après l'heure spécifiée. (En cas de diffusion à flux continu, une recherche vous conduit toujours à l'heure exacte spécifiée même si le fichier FLV source n'y possède pas d'image-clé.) En second lieu, une recherche est asynchrone. Ainsi, si vous appelez une méthode de recherche ou définissez la propriété `playheadTime`, celle-ci n'est pas immédiatement mise à jour. Pour obtenir l'heure à la fin de la recherche, écoutez l'événement `seek`, qui ne se déclenche pas tant que la propriété `playheadTime` n'a pas été mise à jour.

Exemple

L'exemple suivant recherche 2 secondes dans le fichier FLV lorsque l'événement `ready` se produit. La fonction `seek()` déclenche un événement `seek`. A ce moment, l'écouteur affiche la propriété `playheadTime` et le nom de l'occurrence FLVPlayback.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.seek = function(eventObject:Object) {
        trace("A seek event occurred at " + eventObject.playheadTime);
};
my_FLVPlybk.addEventListener("seek", listenerObject);
listenerObject.ready = function(eventObject:Object) {
    my_FLVPlybk.seek(2);
};
my_FLVPlybk.addEventListener("ready", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.activeVideoPlayerIndex`, `FLVPlayback.fastForward`, `FLVPlayback.playheadTime`, `FLVPlayback.playheadUpdate`, `FLVPlayback.rewind`, `FLVPlayback.seek()`, `FLVPlayback.seekPercent()`, `FLVPlayback.seekSeconds()`, `FLVPlayback.seekToNavCuePoint()`, `FLVPlayback.seekToNextNavCuePoint()`, `FLVPlayback.seekToPrevNavCuePoint()`

FLVPlayback.seek()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
my_FLVplybk.seek(time:Number)
```

Paramètres

time Nombre qui indique l'heure, en secondes, à laquelle positionner la tête de lecture.

Renvoie

Aucune.

Description

Méthode : recherche une heure donnée (en secondes) dans le fichier, avec une précision décimale allant jusqu'aux millisecondes.

La propriété `playheadTime` peut ne pas avoir la valeur attendue immédiatement après l'appel aux méthodes de recherche ou la définition de la propriété `playheadTime` pour provoquer une recherche. Il y a là plusieurs raisons. Tout d'abord, s'il s'agit d'un téléchargement progressif, vous pouvez rechercher uniquement une image-clé. Ainsi une recherche vous amène à l'heure de la première image-clé après l'heure spécifiée. (En cas de diffusion à flux continu, une recherche vous conduit toujours à l'heure exacte spécifiée même si le fichier FLV source n'y possède pas d'image-clé.) En second lieu, une recherche est asynchrone. Ainsi si vous appelez une méthode de recherche ou définissez la propriété `playheadTime`, celle-ci n'est pas immédiatement mise à jour. Pour obtenir l'heure à la fin de la recherche, écoutez l'événement `seek`, qui ne se déclenche pas tant que la propriété `playheadTime` n'a pas été mise à jour.

Exemple

L'exemple suivant désactive la lecture automatique du fichier FLV, appelle la méthode `seek()` pour définir les 3 secondes de la tête de lecture dans la vidéo et lance la lecture du fichier FLV à ce stade.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.autoPlay = false;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
my_FLVPlybk.seek(3);
my_FLVPlybk.play();
```

Voir aussi

`FLVPlayback.playheadTime`, `FLVPlayback.seek`, `FLVPlayback.seekPercent()`, `FLVPlayback.seekSeconds()`

FLVPlayback.seekBar

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`my_FLVPlybk``.seekBar`

Description

Propriété : objet MovieClip qui correspond à la commande de la barre de recherche lors de la lecture. Pour plus d'informations sur l'utilisation des composants de l'interface utilisateur personnalisée Lecture de fichiers FLV pour les commandes de lecture, reportez-vous à « Application d'enveloppe aux composants particuliers de l'interface utilisateur personnalisée Lecture de fichiers FLV », à la page 544.

Exemple

L'exemple suivant utilise les propriétés `backButton`, `forwardButton`, `playButton`, `pauseButton`, `stopButton` et `seekBar` pour associer des commandes individuelles de l'interface utilisateur personnalisée FLV à un composant FLVPlayback.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Dans l'inspecteur des composants, définissez le paramètre Skin (Enveloppe) sur None. Ajoutez les composants individuels suivants de l'interface utilisateur personnalisée FLV et donnez-leur les noms d'occurrence indiqués entre parenthèses : BackButton (**my_bkbttn**), ForwardButton (**my_fwdbttn**), PlayPauseButton (**my_plypausbttn**), StopButton (**my_stopbttn**) et SeekBar (**my_seekBar**). Ajoutez ensuite les lignes de code suivantes dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
  - FLV Custom UI BackButton, ForwardButton, PlayPauseButton, StopButton and
   SeekBar
     components in the Library
*/
import mx.video.*;
my_FLVPlybk.backButton = my_bkbttn;
my_FLVPlybk.forwardButton = my_fwdbttn;
my_FLVPlybk.playPauseButton = my_plypausbttn;
my_FLVPlybk.stopButton = my_stopbttn;
my_FLVPlybk.seekBar = my_seekBar;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.scrubbing`, `FLVPlayback.scrubFinish`, `FLVPlayback.scrubStart`, `FLVPlayback.seek`

FLVPlayback.seekBarInterval

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.seekBarInterval

Description

Propriété : nombre indiquant la fréquence (en millisecondes) à laquelle la poignée de la barre de recherche est activée lors de la modulation. La valeur par défaut est 250.

Etant donné que cet intervalle est défini suite à un appel à la fonction globale `setInterval()`, la mise à jour ne peut pas démarrer plus fréquemment que la fréquence d'images du fichier FLV. Par exemple, pour la fréquence par défaut de 12 images par seconde, l'intervalle efficace le plus petit que vous pouvez créer est approximativement de 83 millisecondes ou d'une seconde (1 000 millisecondes) divisée par 12.

Exemple

L'exemple suivant réduit la valeur de `seekBarInterval` pour la définir sur 50 millisecondes, puis affiche la valeur de la propriété `playheadTime` si l'utilisateur effectue une modulation.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.seek = function(eventObject:Object):Void {
  if(my_FLVPlybk.scrubbing) {
    my_FLVPlybk.seekBarInterval = 50;
    trace("User is scrubbing at: " + eventObject.playheadTime);
  }
};
my_FLVPlybk.addEventListener("seek", listenerObject);
```

Voir aussi

`FLVPlayback.seekBar`, `FLVPlayback.seekBarScrubTolerance`

FLVPlayback.seekBarScrubTolerance

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`my_FLVPlybk``.seekBarScrubTolerance`

Description

Propriété : nombre indiquant jusqu'où l'utilisateur peut déplacer la poignée de la barre de recherche avant qu'une mise à jour se produise. Cette valeur est indiquée en pourcentage compris entre 1 et 100. La valeur par défaut est 5.

Exemple

L'exemple suivant vérifie si l'utilisateur effectue une modulation lorsqu'un événement `seek` se produit. Le cas échéant, il diminue la valeur de la propriété `seekBarScrubTolerance` en la définissant sur 0 pour augmenter la mise à jour de l'emplacement de la barre de recherche et la fréquence de l'événement `seek`.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

REMARQUE	Vous devez saisir la poignée de la barre de recherche, la faire glisser, puis la relâcher pour provoquer l'événement.

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.seek = function(eventObject:Object):Void {
  if(my_FLVPlybk.scrubbing) {
    my_FLVPlybk.seekBarScrubTolerance = 0;
    trace("User is scrubbing at: " + eventObject.playheadTime);
  }
};
my_FLVPlybk.addEventListener("seek", listenerObject);
```

Voir aussi

`FLVPlayback.scrubbing`, `FLVPlayback.scrubFinish`, `FLVPlayback.scrubStart`, `FLVPlayback.seekBar`, `FLVPlayback.seekBarInterval`

FLVPlayback.SEEKING

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
mx.video.FLVPlayback.SEEKING
```

Description

Propriété de la classe FLVPlayback en lecture seule qui contient la constante de type chaîne « `seeking` ». Vous pouvez comparer cette propriété avec la propriété `state` pour déterminer si le composant est en état de recherche.

Exemple

L'exemple suivant utilise la propriété `FLVPlayback.SEEKING` pour voir si l'état est bien « `seeking` » lorsqu'un événement `stateChange` se produit. Le cas échéant, il affiche un message le signalant.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  if(eventObject.state == FLVPlayback.SEEKING)
    trace("The current state is " + FLVPlayback.SEEKING);
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
```

Voir aussi

`FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.seekPercent()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
my_FLVplybk.seekPercent(percent:Number)
```

Paramètres

percent Nombre indiquant un pourcentage de la longueur du fichier FLV auquel placer la tête de lecture.

Renvoie

Aucune.

Description

Méthode : recherche un pourcentage du fichier et y positionne la tête de lecture.
Ce pourcentage est un nombre compris entre 0 et 100.

La propriété `playheadTime` peut ne pas avoir la valeur attendue immédiatement après l'appel aux méthodes de recherche ou la définition de la propriété `playheadTime` pour provoquer une recherche. Il y a là plusieurs raisons. Tout d'abord, s'il s'agit d'un téléchargement progressif, vous pouvez rechercher uniquement une image-clé. Ainsi une recherche vous amène à l'heure de la première image-clé après l'heure spécifiée. (En cas de diffusion à flux continu, une recherche vous conduit toujours à l'heure exacte spécifiée même si le fichier FLV source n'y possède pas d'image-clé.) En second lieu, une recherche est asynchrone. Ainsi si vous appelez une méthode de recherche ou définissez la propriété `playheadTime`, celle-ci n'est pas immédiatement mise à jour. Pour obtenir l'heure à la fin de la recherche, écoutez l'événement `seek`, qui ne se déclenche pas tant que la propriété `playheadTime` n'a pas été mise à jour.

Exemple

L'exemple suivant désactive la lecture automatique du fichier FLV. Si le fichier FLV est prêt, il définit 30 pour cent de tête de lecture dans le temps de lecture, et commence la lecture à ce stade.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.autoPlay = false;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object) {
  my_FLVPlybk.seekPercent(30);
  my_FLVPlybk.play();
}
my_FLVPlybk.addEventListener("ready", listenerObject);
```

Voir aussi

FLVPlayback.seek, FLVPlayback.seek(), FLVPlayback.seekSeconds()

FLVPlayback.seekSeconds()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVplybk.seekSeconds(*time*:Number)

Paramètres

time Nombre indiquant le temps (en secondes) de la durée totale de lecture auquel placer la tête de lecture.

Renvoie

Aucune.

Description

Méthode : recherche une heure donnée (en secondes) dans le fichier, avec une précision décimale allant jusqu'aux millisecondes. Cette méthode effectue la même opération que la méthode `seek()` ; elle est fournie pour la symétrie avec la méthode `seekPercent()`.

La propriété `playheadTime` peut ne pas avoir la valeur attendue immédiatement après l'appel aux méthodes de recherche ou la définition de la propriété `playheadTime` pour provoquer une recherche. Il y a là plusieurs raisons. Tout d'abord, s'il s'agit d'un téléchargement progressif, vous pouvez rechercher uniquement une image-clé. Ainsi une recherche vous amène à l'heure de la première image-clé après l'heure spécifiée. (En cas de diffusion à flux continu, une recherche vous conduit toujours à l'heure exacte spécifiée même si le fichier FLV source n'y possède pas d'image-clé.) En second lieu, une recherche est asynchrone. Ainsi si vous appelez une méthode de recherche ou définissez la propriété `playheadTime`, celle-ci n'est pas immédiatement mise à jour. Pour obtenir l'heure à la fin de la recherche, écoutez l'événement `seek`, qui ne se déclenche pas tant que la propriété `playheadTime` n'a pas été mise à jour.

Exemple

L'exemple suivant désactive la lecture automatique du fichier FLV, appelle la méthode `seekSeconds()` pour définir les 5 secondes de la tête de lecture dans la vidéo et commence la lecture du fichier FLV à ce stade.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.autoPlay = false;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object) {
  my_FLVPlybk.seekSeconds(4);
  my_FLVPlybk.play();
}
my_FLVPlybk.addEventListener("ready", listenerObject);
```

Voir aussi

`FLVPlayback.seek`, `FLVPlayback.seek()`, `FLVPlayback.seekPercent()`

FLVPlayback.seekToNavCuePoint()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
my_FLVplybk.seekToNavCuePoint(time:Number):Void
my_FLVplybk.seekToNavCuePoint(name:String):Void
my_FLVplybk.seekToNavCuePoint(cuePoint:Object):Void
```

Paramètres

time Nombre indiquant l'heure du point de repère de navigation à rechercher. La méthode utilise uniquement les trois premiers chiffres et arrondit tout chiffre supplémentaire.

name Chaîne contenant le nom du point de repère à rechercher.

cuePoint Objet point de repère dans lequel vous définissez les propriétés `time` et `name` pour spécifier le point de repère à rechercher.

Renvoie

Aucune.

Description

Méthode : recherche un point de repère de navigation correspondant à l'heure spécifiée ou ultérieur à cette heure. Si l'heure n'est pas définie, si sa valeur est `null` ou négative, cette méthode démarre la recherche à l'heure 0.

Si vous ne spécifiez qu'une heure, la méthode recherche un point de repère correspondant à cette heure ou ultérieur.

Si vous indiquez un nom, la méthode recherche le premier point de repère activé correspondant. Pour plus d'informations sur l'activation et la désactivation des points de repère, reportez-vous à « FLVPlayback.setFLVCuePointEnabled() », à la page 691.

La propriété `playheadTime` peut ne pas avoir la valeur attendue immédiatement après l'appel aux méthodes de recherche ou la définition de la propriété `playheadTime` pour provoquer une recherche. Il y a là plusieurs raisons. Tout d'abord, s'il s'agit d'un téléchargement progressif, vous pouvez rechercher uniquement une image-clé. Ainsi une recherche vous amène à l'heure de la première image-clé après l'heure spécifiée. (En cas de diffusion à flux continu, une recherche vous conduit toujours à l'heure exacte spécifiée même si le fichier FLV source n'y possède pas d'image-clé.) En second lieu, une recherche est asynchrone. Ainsi si vous appelez une méthode de recherche ou définissez la propriété `playheadTime`, celle-ci n'est pas immédiatement mise à jour. Pour obtenir l'heure à la fin de la recherche, écoutez l'événement `seek`, qui ne se déclenche pas tant que la propriété `playheadTime` n'a pas été mise à jour.

Exemple

L'exemple suivant recherche le point de repère `point2` lorsque l'événement `ready` se produit. Le gestionnaire d'événements `cuePoint` affiche les valeurs `name`, `time` et `type` de chaque point de repère trouvé.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  my_FLVPlybk.seekToNavCuePoint("point2");
}
my_FLVPlybk.addEventListener("ready", listenerObject);
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object):Void {
    trace("Cue point name is: " + eventObject.info.name);
    trace("Cue point time is: " + eventObject.info.time);
    trace("Cue point type is: " + eventObject.info.type);
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Voir aussi

`FLVPlayback.cuePoint`, `FLVPlayback.seek`, `FLVPlayback.seek()`, `FLVPlayback.seekToNextNavCuePoint()`

FLVPlayback.seekToNextNavCuePoint()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVplybk.seekToNextNavCuePoint([*time*:Number])

Paramètres

time Nombre représentant l'heure de début (en secondes) à partir de laquelle rechercher le point de repère de navigation suivant. Le nombre par défaut est la valeur actuelle de la propriété `playheadTime`. Facultatif.

Renvoie

Aucune.

Description

Méthode : recherche le point de repère de navigation suivant, en fonction de la valeur actuelle de la propriété `playheadTime`. Cette méthode ignore les points de repère de navigation désactivés et accède à la fin du fichier FLV s'il n'y a pas d'autres points de repère.

La propriété `playheadTime` peut ne pas avoir la valeur attendue immédiatement après l'appel aux méthodes de recherche ou la définition de la propriété `playheadTime` pour provoquer une recherche. Il y a là plusieurs raisons. Tout d'abord, s'il s'agit d'un téléchargement progressif, vous pouvez rechercher uniquement une image-clé. Ainsi une recherche vous amène à l'heure de la première image-clé après l'heure spécifiée. (En cas de diffusion à flux continu, une recherche vous conduit toujours à l'heure exacte spécifiée même si le fichier FLV source n'y possède pas d'image-clé.) En second lieu, une recherche est asynchrone. Ainsi si vous appelez une méthode de recherche ou définissez la propriété `playheadTime`, celle-ci n'est pas immédiatement mise à jour. Pour obtenir l'heure à la fin de la recherche, écoutez l'événement `seek`, qui ne se déclenche pas tant que la propriété `playheadTime` n'a pas été mise à jour.

Exemple

L'exemple suivant recherche le point de repère de navigation suivant lorsque le point de repère `point2` est trouvé. La partie du fichier FLV située entre les points de repère `point2` et `point3` est alors ignorée.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object) {
     if(eventObject.info.name == "point2")
       my_FLVPlybk.seekToNextNavCuePoint(eventObject.info.time);
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Voir aussi

`FLVPlayback.cuePoint`, `FLVPlayback.findCuePoint()`, `FLVPlayback.playheadTime`, `FLVPlayback.seekToNavCuePoint()`, `FLVPlayback.seekToPrevNavCuePoint()`, `FLVPlayback.isFLVCuePointEnabled()`, `FLVPlayback.setFLVCuePointEnabled()`

FLVPlayback.seekToPrevNavCuePoint()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`my_FLVplybk``.seekToPrevNavCuePoint([`*`time`*`:Number])`

Paramètres

`time` Nombre représentant l'heure de début (en secondes) à partir de laquelle rechercher le point de repère de navigation précédent. La valeur par défaut est la valeur actuelle de la propriété `playheadTime`. Facultatif.

Renvoie

Aucune.

Description

Méthode : recherche le point de repère de navigation précédent, en fonction de la valeur actuelle de la propriété `playheadTime`. Elle accède au début du fichier FLV s'il n'existe pas de points de repère précédents. La méthode ignore les points de repère de navigation qui ont été désactivés.

La propriété `playheadTime` peut ne pas avoir la valeur attendue immédiatement après l'appel aux méthodes de recherche ou la définition de la propriété `playheadTime` pour provoquer une recherche. Il y a là plusieurs raisons. Tout d'abord, s'il s'agit d'un téléchargement progressif, vous pouvez rechercher uniquement une image-clé. Ainsi une recherche vous amène à l'heure de la première image-clé après l'heure spécifiée. (En cas de diffusion à flux continu, une recherche vous conduit toujours à l'heure exacte spécifiée même si le fichier FLV source n'y possède pas d'image-clé.) En second lieu, une recherche est asynchrone. Ainsi si vous appelez une méthode de recherche ou définissez la propriété `playheadTime`, celle-ci n'est pas immédiatement mise à jour. Pour obtenir l'heure à la fin de la recherche, écoutez l'événement `seek`, qui ne se déclenche pas tant que la propriété `playheadTime` n'a pas été mise à jour.

Exemple

L'exemple suivant recherche le point de repère de navigation précédent lorsque le point de repère, point2, est trouvé, ce qui crée une boucle de lecture du fichier FLV.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object) {
    if(eventObject.info.name == "point2")
      my_FLVPlybk.seekToPrevNavCuePoint(eventObject.info.time);
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject);
```

Voir aussi

`FLVPlayback.cuePoint`, `FLVPlayback.findCuePoint()`, `FLVPlayback.playheadTime`, `FLVPlayback.seekToNavCuePoint()`, `FLVPlayback.seekToPrevNavCuePoint()`, `FLVPlayback.isFLVCuePointEnabled()`, `FLVPlayback.setFLVCuePointEnabled()`

FLVPlayback.seekToPrevOffset

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.seekToPrevOffset

Description

Propriété : nombre (en secondes) utilisé par la méthode `seekToPrevNavCuePoint()` lors de la comparaison de son heure avec celle du point de repère précédent. La méthode utilise cette valeur afin d'assurer que si vous vous trouvez devant un point de repère, vous pouvez l'ignorer pour passer au précédent sans accéder au même point de repère. Elle est définie par défaut sur une seconde.

Exemple

L'exemple suivant définit au début la propriété `seekToPrevOffset` sur 10. Le premier appel à la méthode `seekToPrevNavCuePoint()` trouve donc le point de repère, point1. Lorsque l'événement `cuePoint` initial du point de repère point3 se produit, l'exemple diminue cependant la valeur de la propriété `seekToPrevOffset` sur 1 seconde. Les appels suivants à la méthode `seekToPrevNavCuePoint()` atteignent donc le point de repère, point2.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object) {
  my_FLVPlybk.seekToPrevOffset = 10;
  my_FLVPlybk.seekToNavCuePoint("point3");
}
```

```
my_FLVPlybk.addEventListener("ready", listenerObject)
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object) {
    trace("hit cue point at " + eventObject.info.time);
    if(eventObject.info.name == "point3"){
      my_FLVPlybk.seekToPrevNavCuePoint(eventObject.info.time);
      my_FLVPlybk.seekToPrevOffset = 1;
    }
}
my_FLVPlybk.addEventListener("cuePoint", listenerObject)
```

Voir aussi

FLVPlayback.seekToPrevNavCuePoint()

FLVPlayback.setFLVCuePointEnabled()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
my_FLVplybk.setFLVCuePointEnabled(enabled:Boolean, time:Number)
my_FLVplybk.setFLVCuePointEnabled(enabled:Boolean, name:String)
my_FLVplybk.setFLVCuePointEnabled(enabled:Boolean, cuePoint:Object)
```

Paramètres

enabled Valeur booléenne indiquant si un point de repère FLV doit être activé (`true`) ou non (`false`)

time Nombre correspondant à l'heure, en secondes, du point de repère à définir.

name Nom du point de repère à définir.

cuePoint Objet point de repère possédant les propriétés `time` et `name` correspondant au point de repère à définir. Cette méthode n'active aucune autre propriété sur l'objet point de repère entrant. Si le paramètre `time` ou `name` a la valeur undefined, la méthode tente de faire correspondre un point de repère en utilisant la valeur disponible uniquement.

Renvoie

Un nombre. Si le paramètre `metadataLoaded` a la valeur `true`, la méthode renvoie le nombre de points de repère dont l'état activé a été modifié. Si le paramètre `metadataLoaded` est défini sur `false`, la méthode renvoie la valeur 1, car le composant ne peut toujours pas déterminer les points de repére à définir, s'il en existe. Lorsque les métadonnées sont chargées, le composant définit cependant les points de repère spécifiés de façon appropriée.

Description

Méthode : active ou désactive un ou plusieurs points de repère de fichier FLV. Les points de repère sont désactivés pour pouvoir être distribués en tant qu'événements et pour que vous puissiez y accéder avec les méthodes `seekToPrevNavCuePoint()`, `seekToNextNavCuePoint()` et `seekToNavCuePoint()`.

Les informations sur les points de repère sont supprimées lorsque vous définissez la propriété `contentPath` sur un autre fichier FLV. Vous devez donc définir cette propriété avant les informations sur les points de repère relatives au fichier FLV suivant à charger.

Les changements provoqués par cette fonction ne sont pas reflétés par les appels à la méthode `isFLVCuePointEnabled()` tant que les métadonnées n'ont pas été chargées.

Exemple

L'exemple suivant désactive les points de repère `point2` et `point3` lorsque l'événement `ready` se produit. Le gestionnaire d'événements `cuePoint` affiche dans le panneau Sortie le nom et l'heure de chaque point de repère trouvé. Le fichier FLV contient les points de repère intégrés suivants : point1 à 00:00:00:418 ; point2 à 00:00:07.748 ; point3 à 00:00:16:020.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
function ready(eventObject:Object) {
  my_FLVPlybk.setFLVCuePointEnabled(false, "point2");
  my_FLVPlybk.setFLVCuePointEnabled(false, 16.02);
}
my_FLVPlybk.addEventListener("ready", ready);
function cuePoint(eventObject:Object) {
    trace("Cue point name is: " + eventObject.info.name);
    trace("Cue point time is: " + eventObject.info.time);
}
my_FLVPlybk.addEventListener("cuePoint", cuePoint);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Voir aussi

FLVPlayback.cuePoint, FLVPlayback.findCuePoint(), FLVPlayback.findNearestCuePoint(), FLVPlayback.findNextCuePointWithName(), FLVPlayback.isFLVCuePointEnabled(), FLVPlayback.seekToNavCuePoint(), FLVPlayback.seekToNextNavCuePoint(), FLVPlayback.seekToPrevNavCuePoint()

FLVPlayback.setScale()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
my_FLVplybk.setScale(xs:Number, ys:Number)
```

Paramètres

xs Nombre représentant le redimensionnement horizontal.

ys Nombre représentant le redimensionnement vertical.

Renvoie

Aucune.

Description

Méthode : définit simultanément les propriétés `scaleX` et `scaleY`. Comme la définition de l'une ou l'autre de ces propriétés peut provoquer un redimensionnement automatique, il peut être plus efficace de les définir simultanément.

Si le paramètre `autoSize` a la valeur `true`, cette méthode n'a aucune incidence, car le lecteur définit ses propres dimensions. Si la propriété `maintainAspectRatio` a la valeur `true` et si la propriété `autoSize` a la valeur `false`, la modification des propriétés `scaleX` ou `scaleY` peut provoquer un redimensionnement automatique.

Exemple

L'exemple suivant appelle la méthode `setScale()` pour redimensionner les axes horizontal (*x*) et vertical (*y*) de l'occurrence FLVPlayback. Cet exemple définit la propriété `maintainAspectRatio` sur `false` pour empêcher le redimensionnement automatique et permettre celui que vous avez spécifié.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;

my_FLVPlybk.maintainAspectRatio = false;
my_FLVPlybk.setScale(200, 175);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.scaleX`, `FLVPlayback.scaleY`, `FLVPlayback.setSize()`

FLVPlayback.setSize()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVplybk.setSize(*w*:Number, *h*:Number)

Paramètres

w Nombre indiquant la largeur du lecteur vidéo.

h Nombre indiquant la hauteur du lecteur vidéo.

Renvoie

Aucune.

Description

Méthode : définit simultanément les propriétés `width` et `height`. Comme la définition de l'une ou l'autre de ces propriétés peut provoquer un redimensionnement automatique, il peut être plus efficace de les définir simultanément.

Si le paramètre `autoSize` a la valeur `true`, cette méthode n'a aucune incidence, car le lecteur définit ses propres dimensions. Si la propriété `maintainAspectRatio` a la valeur `true` et si la propriété `autoSize` a la valeur `false`, la modification des propriétés `width` ou `height` peut provoquer un redimensionnement automatique.

Exemple

L'exemple suivant appelle la méthode `setSize()` pour définir la taille de l'occurrence FLVPlayback sur une largeur de 150 pixels et une hauteur de 150 pixels. Le gestionnaire d'événements `resize` affiche la largeur et la hauteur réelles, car la propriété `maintainAspectRatio` est définie par défaut sur `true`. Ainsi un redimensionnement automatique conserve les proportions.

Faites glisser le composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybkh
*/
import mx.video.*;
// Propriété maintainAspectRatio définie sur true par défaut.
// Les dimensions conservent donc leurs proportions.
my_FLVPlybk.setSize(150, 150);
var listenerObject:Object = new Object();
listenerObject.resize = function(eventObject:Object):Void {
  trace("Player's width is: " + my_FLVPlybk.width)
  trace("Player's height is: " + my_FLVPlybk.height)
};
my_FLVPlybk.addEventListener("resize", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.height`, `FLVPlayback.width`, `FLVPlayback.setScale()`

FLVPlayback.skin

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.skin

Description

Propriété : chaîne indiquant l'URL d'un fichier SWF d'enveloppe. Cette chaîne peut contenir un nom de fichier, un chemin relatif tel que FLVPlayback FLVPlayback Skins/ActionScript 2.0/my_Skin.swf, ou une URL absolue comme http://www.myskins.org/MySkin.swf.

Exemple

L'exemple suivant applique l'enveloppe ArcticExternal.swf à une occurrence du composant FLVPlayback.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Copiez le fichier ArcticExternalAll.swf du dossier Flash CS3 Configuration/FLVPlayback Skins/ActionScript 2.0 vers votre dossier de travail. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybkh
*/
import mx.video.*;
my_FLVPlybk.skin = "ArcticExternalAll.swf";
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

FLVPlayback.bufferingBarHidesAndDisablesOthers, FLVPlayback.skinAutoHide, FLVPlayback.skinError, FLVPlayback.skinLoaded

FLVPlayback.skinAutoHide

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVplybk.skinAutoHide

Description

Propriété ; valeur booléenne qui, si elle est définie sur `true`, masque l'enveloppe du composant lorsque la souris n'est pas placée sur la vidéo. Cette propriété affecte uniquement les enveloppes chargées en définissant la propriété `skin` et pas celles que vous créez dans les composants de l'interface utilisateur personnalisée Lecture de fichiers FLV. La valeur par défaut est `false`.

Exemple

L'exemple suivant définit la propriété `skinAutoHide` sur `true` pour que le composant Skin, qui inclut les commandes de lecture, n'apparaisse pas à moins que la souris soit posée sur la vidéo.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Sélectionnez une enveloppe dans l'inspecteur de composants. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybkh
*/
import mx.video.*;
my_FLVPlybk.skinAutoHide = true;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.bufferingBarHidesAndDisablesOthers`, `FLVPlayback.skin`

FLVPlayback.skinError

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.skinError = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("skinError", listenerObject);
```

Description

Evénement : distribué lorsqu'une erreur se produit lors du chargement d'un fichier SWF d'enveloppe. Cet événement possède une propriété `message` qui contient le message d'erreur.

Exemple

L'exemple suivant tente de charger la propriété `skin` avec le nom d'un fichier d'enveloppe fictif et affiche le contenu de la propriété `message` de l'événement lorsque l'événement `skinError` se produit.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez ensuite le code suivant à l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybkh
*/
import mx.video.*;

var listenerObject:Object = new Object();
listenerObject.skinError = function(eventObject:Object):Void {
  trace(eventObject.message);
}
my_FLVPlybk.addEventListener("skinError", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
my_FLVPlybk.skin = "NoSuchSkin.swf";
```

Voir aussi

`FLVPlayback.skin`, `FLVPlayback.skinLoaded`

FLVPlayback.skinLoaded

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.skinLoaded = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("skinLoaded", listenerObject);
```

Description

Evénement : distribué lors du chargement d'un fichier SWF d'enveloppe. Le composant ne commence pas à lire un fichier FLV tant que les deux événements `ready` et `skinLoaded` (ou `skinError`) ne se sont pas produits.

Exemple

L'exemple suivant affiche le nom de l'enveloppe du composant lorsque l'événement `skinLoaded` se produit.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez ensuite le code suivant à l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybkh
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.skinLoaded = function(eventObject:Object):Void {
  trace("Skin: " + eventObject.target.skin + " has loaded");
};
my_FLVPlybk.addEventListener("skinLoaded", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Voir aussi

`FLVPlayback.addEventListener()`, `FLVPlayback.skin`, `FLVPlayback.skinError`

FLVPlayback.state

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`my_FLVPlybk``.state`

Description

Propriété : nombre indiquant l'état du composant. Cette propriété est définie avec les méthodes `load()`, `play()`, `stop()`, `pause()` et `seek()`. Lecture seule.

La propriété `state` peut avoir les valeurs suivantes : « `buffering` », « `connectionError` », « `disconnected` », « `loading` », « `paused` », « `playing` », « `rewinding` », « `seeking` » et « `stopped` ». Vous pouvez utiliser les propriétés de la classe FLVPlayback pour tester ces états. Pour plus d'informations, voir « Propriétés de la classe FLVPlayback », à la page 562.

Exemple

L'exemple suivant affiche la propriété `state` dans le panneau Sortie à chaque fois que l'événement `stateChange` se produit lors de la lecture du fichier FLV.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  trace(my_FLVPlybk.state);
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.addEventListener()`, `FLVPlayback.buffering`, `FLVPlayback.paused`, `FLVPlayback.playing`, `FLVPlayback.stateChange`, `FLVPlayback.stopped`

FLVPlayback.stateChange

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("stateChange", listenerObject);
```

Description

Evénement ; distribué lorsque l'état de lecture est modifié. L'objet événement a les propriétés `state` et `playheadTime`.

Cet événement peut être utilisé pour suivre le moment où la lecture entre/quitte des états non réactifs (par exemple au milieu d'une connexion, d'un redimensionnement ou d'un rembobinage) pendant lesquels les méthodes `play()`, `pause()`, `stop()` et `seek()` mettent en file d'attente les requêtes à exécuter lorsque le lecteur entre dans un état réactif.

L'événement possède la propriété `vp`, qui correspond au numéro d'index du lecteur vidéo auquel cet événement s'applique. Pour plus d'informations sur la propriété `vp`, reportez-vous à « FLVPlayback.activeVideoPlayerIndex » à la page 571 et à « FLVPlayback.visibleVideoPlayerIndex » à la page 713.

Exemple

L'exemple suivant affiche la propriété `state` dans le panneau Sortie à chaque fois que l'événement `stateChange` se produit lors de la lecture du fichier FLV.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  trace(my_FLVPlybk.state);
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.addEventListener()`, `FLVPlayback.state`

FLVPlayback.stateResponsive

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.stateResponsive

Description

Propriété : valeur booléenne définie sur `true` si l'état est réactif. Si l'état n'est pas réactif, les appels aux méthodes `play()`, `load()`, `stop()`, `pause()` et `seek()` sont placés en file d'attente et exécutés plus tard lorsque l'état change et devient réactif. Etant donné que ces appels sont mis en file d'attente pour être exécutés plus tard, il n'est habituellement pas nécessaire de suivre la valeur de la propriété `stateResponsive`. Les états réactifs sont les suivants : `disconnected`, `stopped`, `playing`, `paused` et `buffering`. Lecture seule.

Exemple

L'exemple suivant affiche les valeurs des propriétés `state` et `stateResponsive` à mesure que l'état change pendant la lecture du fichier FLV.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  trace(my_FLVPlybk.state + "; responsive: " +
  my_FLVPlybk.stateResponsive);
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.stop()

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.stop()

Paramètres

Aucun.

Renvoie

Aucune.

Description

Méthode : arrête la lecture de la vidéo. Si la propriété `autoRewind` a la valeur `true`, le fichier FLV se rembobine au début.

Exemple

L'exemple suivant écoute l'événement `playheadUpdate` et lorsque la propriété `playheadTime` est supérieure ou égale à 5 secondes, l'écouteur appelle la méthode `stop()` pour arrêter la lecture du fichier FLV. Un second écouteur écoute l'événement `stopped` et affiche les valeurs des propriétés `playheadTime` et `state`.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.autoRewind = false;
var listenerObject:Object = new Object();
listenerObject.stopped = function(eventObject:Object):Void {
  trace("playhead time is: " + eventObject.playheadTime);
  trace("The video player state is: " + eventObject.state);
};
```

```
my_FLVPlybk.addEventListener("stopped", listenerObject);
listenerObject.playheadUpdate = function(eventObject:Object):Void {
  if (eventObject.playheadTime >= 5) {
    my_FLVPlybk.stop();
  }
};
my_FLVPlybk.addEventListener("playheadUpdate", listenerObject);
my_FLVPlybk.contentPath ="http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.pause()`, `FLVPlayback.play()`

FLVPlayback.stopButton

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.stopButton

Description

Propriété : objet MovieClip qui correspond à la commande du bouton Arrêt. Pour plus d'informations sur l'utilisation des composants de l'interface utilisateur personnalisée Lecture de fichiers FLV pour les commandes de lecture, reportez-vous à « Application d'enveloppe aux composants particuliers de l'interface utilisateur personnalisée Lecture de fichiers FLV », à la page 544.

Exemple

L'exemple suivant utilise les propriétés `backButton`, `forwardButton`, `playButton`, `pauseButton` et `stopButton` pour associer des commandes individuelles de l'interface utilisateur personnalisée Lecture de fichiers FLV à un composant FLVPlayback.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Dans l'inspecteur des composants, définissez le paramètre Skin (Enveloppe) sur None. Ajoutez les composants individuels suivants de l'interface utilisateur personnalisée Lecture de fichiers FLV et donnez-leur les noms d'occurrence indiqués entre parenthèses : BackButton (**my_bkbttn**), ForwardButton (**my_fwdbttn**), PlayButton (**my_plybttn**), PauseButton (**my_pausbttn**) et StopButton (**my_stopbttn**). Ajoutez ensuite les lignes de code suivantes au panneau Actions :

```
/**
 Requires:
  - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
  - FLV Custom UI BackButton, ForwardButton, PlayButton, PauseButton, and
    StopButton components in the Library
*/
import mx.video.*;
my_FLVPlybk.backButton = my_bkbttn;
my_FLVPlybk.forwardButton = my_fwdbttn;
my_FLVPlybk.playButton = my_plybttn;
my_FLVPlybk.pauseButton = my_pausbttn;
my_FLVPlybk.stopButton = my_stopbttn;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.pauseButton`, `FLVPlayback.playButton`, `FLVPlayback.playPauseButton`, `FLVPlayback.skin`, `FLVPlayback.stopped`

FLVPlayback.STOPPED

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`mx.video.FLVPlayback.STOPPED`

Description

Propriété de la classe FLVPlayback en lecture seule qui contient la constante de type chaîne « `stopped` ». Vous pouvez comparer cette propriété avec la propriété `state` pour déterminer si le composant est en état d'arrêt.

Exemple

L'exemple suivant affiche la valeur de la propriété `FLVPlayback.STOPPED` lorsque le composant entre dans un état d'arrêt.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  if(eventObject.state == FLVPlayback.STOPPED)
    trace("State is " + FLVPlayback.STOPPED);
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.state`, `FLVPlayback.stateChange`

FLVPlayback.stopped

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.stopped = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("stopped", listenerObject);
```

Description

Evénement : distribué lorsque le composant entre dans l'état d'arrêt. Ceci se produit lorsque vous appelez la méthode `stop()` ou que vous cliquez sur la commande `stopButton`. Cela se produit également dans certains cas, si la propriété `autoPlay` a la valeur `false` (l'état peut prendre la valeur `paused`) lorsque le fichier FLV est chargé. L'occurrence FLVPlayback distribue également cet événement lorsque la tête de lecture s'arrête à la fin du fichier FLV. L'objet événement possède les propriétés `state`, `playheadTime` et `vp`, qui correspond au numéro d'index du lecteur vidéo auquel cet événement s'applique. Pour plus d'informations sur la propriété `vp`, reportez-vous à « FLVPlayback.activeVideoPlayerIndex » à la page 571 et à « FLVPlayback.visibleVideoPlayerIndex » à la page 713.

L'occurrence de composant FLVPlayback distribue également l'événement `stateChange`.

Exemple

L'exemple suivant écoute les occurrences de l'événement `stopped` lorsqu'il a lieu pendant la lecture du fichier FLV. Lorsque cet événement a lieu, l'exemple affiche la durée de lecture écoulée dans le panneau Sortie.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stopped = function(eventObject:Object):Void {
  trace(my_FLVPlybk.state + ": playhead time is: " +
  eventObject.playheadTime);
};
my_FLVPlybk.addEventListener("stopped", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.addEventListener()`, `FLVPlayback.playheadTime`, `FLVPlayback.state`, `FLVPlayback.stateChange`, `FLVPlayback.stop()`

FLVPlayback.stopped

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.stopped

Description

Propriété : valeur booléenne définie sur `true` si l'état de l'occurrence FLVPlayback est `stopped`. Lecture seule.

Exemple

L'exemple suivant écoute les occurrences de l'événement `stateChange` lorsqu'il a lieu pendant la lecture du fichier FLV. Lorsque cet événement a lieu, l'exemple affiche la valeur de la propriété `stopped` dans le panneau Sortie.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  trace(my_FLVPlybk.state + ": stopped property is: " +
  my_FLVPlybk.stopped);
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.state`, `FLVPlayback.stateChange`, `FLVPlayback.stop()`, `FLVPlayback.stopped`

FLVPlayback.totalTime

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.totalTime

Description

Propriété : nombre représentant la durée totale de lecture de la vidéo en secondes. Lors de la diffusion en continu à partir d'un FMS et de l'utilisation de NCManager par défaut, cette valeur est déterminée automatiquement par des API côté serveur, et cette valeur écrase celle définie au moyen de cette propriété ou obtenue à partir de métadonnées. Cela vaut également si vous définissez cette valeur dans un fichier SMIL. La propriété est prête pour la lecture lorsque l'état d'arrêt ou de lecture est atteint après la définition de la propriété `contentPath`. Cette propriété n'a aucune signification pour les flux en direct à partir d'un FMS.

S'il s'agit d'un téléchargement HTTP, la valeur est déterminée automatiquement si des métadonnées sont intégrées au fichier FLV. Autrement, vous devez la définir de façon explicite ou elle sera égale à 0. Si vous la définissez, la valeur des métadonnées dans le flux est ignorée.

Lorsque vous définissez cette propriété, la valeur est effective pour le prochain fichier FLV chargé en définissant `contentPath`. Elle n'a aucun effet sur un fichier FLV déjà chargé. En outre, cette propriété ne renvoie pas la nouvelle valeur transmise tant qu'un fichier FLV n'a pas été chargé.

La lecture continue si cette propriété n'est jamais définie (de façon explicite ou automatique), mais elle risque de poser des problèmes avec les commandes de recherche.

Exemple

L'exemple suivant affiche la durée totale du fichier FLV (en secondes) lorsque l'événement `ready` a lieu, à la fin du chargement.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  trace("Total play time for this video is: " + my_FLVPlybk.totalTime);
};
my_FLVPlybk.addEventListener("ready", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.contentPath`, `FLVPlayback.playheadTime`, `FLVPlayback.playing`, `FLVPlayback.stopped`

FLVPlayback.transform

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`my_FLVPlybk``.transform`

Description

Propriété : objet permettant d'accéder directement aux méthodes `Sound.setTransform()` et `Sound.getTransform()` afin de fournir un meilleur son. Vous devez définir cette propriété sur un objet pour l'initialiser et pour que les modifications prennent effet. La lecture de la propriété vous fournit une copie des paramètres actuels que vous pouvez modifier. La valeur par défaut est `undefined`.

Exemple

L'exemple suivant définit la propriété `transform` pour lire le son du fichier FLV à partir du haut-parleur gauche uniquement.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
/* Lecture de tout l'audio à partir du haut-parleur gauche uniquement */
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object) {
  if (eventObject.target.state == "loading") {  // En cas de chargement
    myTransform = new Object();
    myTransform.ll = 100;
    myTransform.lr = 100;
    myTransform.rr = 0;
    myTransform.rl = 0;
    my_FLVPlybk.transform = myTransform;
  }
};
my_FLVPlybk.addEventListener("stateChange", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Voir aussi

`FLVPlayback.volume`

FLVPlayback.version

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`mx.video.FLVPlayback.version`

Description

Propriété de la classe FLVPlayback en lecture seule qui contient le numéro de version du composant.

Exemple

L'exemple suivant affiche le numéro de version du composant dans le panneau Sortie. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
import mx.video.*;
trace(FLVPlayback.version);
```

FLVPlayback.visible

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
my_FLVPlybk.visible
```

Description

Propriété ; valeur booléenne qui, si elle est définie sur `true`, affiche le composant FLVPlayback. Si elle a la valeur `false`, le composant est masqué. La valeur par défaut est `true`.

Exemple

L'exemple suivant définit la propriété `visible` sur `false` pour masquer l'occurrence FLVPlayback à la fin de la lecture du fichier FLV.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.complete = function(eventObject:Object):Void {
  my_FLVPlybk.visible = false;
};
my_FLVPlybk.addEventListener("complete", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.activeVideoPlayerIndex`, `FLVPlayback.closeVideoPlayer()`, `FLVPlayback.visibleVideoPlayerIndex`

FLVPlayback.visibleVideoPlayerIndex

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.visibleVideoPlayerIndex

Description

Propriété : nombre que vous pouvez utiliser pour gérer plusieurs flux continus de fichier FLV. Définissez quelle occurrence de lecteur vidéo est visible, audible et contrôlée par les commandes d'enveloppe ou de lecture alors que les autres lecteurs sont masqués et silencieux. La valeur par défaut est 0. Le lecteur vidéo n'est pas la cible pour la plupart des API ; utilisez plutôt la propriété `activeVideoPlayerIndex`.

Les méthodes et les propriétés qui contrôlent les dimensions interagissent avec cette propriété. Les méthodes et les propriétés qui définissent les dimensions du lecteur vidéo (`setScale()`, `setSize()`, `width`, `height`, `scaleX`, `scaleY`) peuvent être appliquées à tous les lecteurs vidéo. Néanmoins, selon que `autoSize` ou `maintaintAspectRatio` est défini sur ces lecteurs vidéo, ils peuvent avoir des dimensions différentes. La lecture des dimensions à l'aide des propriétés `width`, `height`, `scaleX` et `scaleY` vous donne uniquement les dimensions du lecteur vidéo visible. Les autres lecteurs vidéo peuvent ou non avoir les mêmes dimensions.

Pour obtenir les dimensions de différents lecteurs vidéo lorsqu'ils ne sont pas visibles, écoutez l'événement `resize`, puis enregistrez la valeur de leur taille.

Cette propriété n'a aucune implication sur la visibilité du composant dans son ensemble, uniquement sur le lecteur vidéo visible lorsque le composant est visible. Pour définir la visibilité pour le composant entier, utilisez la propriété `visible`.

Exemple

L'exemple suivant crée deux lecteurs vidéo pour lire deux fichiers FLV de manière consécutive dans une même occurrence FLVPlayBack. Il définit la propriété `visibleVideoPlayerIndex` pour rendre les lecteurs vidéo et les fichiers FLV visibles.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
// Indication du nom et de l'emplacement du fichier FLV pour le lecteur
// par défaut.
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  clouds.flv"
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  // Ajout d'un second lecteur vidéo et indication du nom et
  // de l'emplacement de son fichier FLV.
  my_FLVPlybk.activeVideoPlayerIndex = 1;
  my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
  // Réinitialisation sur le lecteur vidéo par défaut qui lit
  // automatiquement son fichier FLV.
  my_FLVPlybk.activeVideoPlayerIndex = 0;
};
my_FLVPlybk.addEventListener("ready", listenerObject);
listenerObject.complete = function(eventObject:Object):Void {
  // Si l'événement complete concerne le second fichier FLV,
  // rendez-le actif et visible par défaut.
  if (eventObject.vp == 1) {
    my_FLVPlybk.activeVideoPlayerIndex = 0;
    my_FLVPlybk.visibleVideoPlayerIndex = 0;
  } else { // Activation et affichage du second lecteur, lecture du fichier FLV.
    my_FLVPlybk.activeVideoPlayerIndex = 1;
    my_FLVPlybk.visibleVideoPlayerIndex = 1;
    my_FLVPlybk.play();
  }
};
// Ajout d'un écouteur pour un événement complete.
my_FLVPlybk.addEventListener("complete", listenerObject);
```

Voir aussi

FLVPlayback.activeVideoPlayerIndex, FLVPlayback.setScale(), FLVPlayback.setSize(), FLVPlayback.height, FLVPlayback.width, FLVPlayback.scaleX, FLVPlayback.scaleY, FLVPlayback.visible

FLVPlayback.volume

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.volume

Description

Propriété ; chiffre compris entre 0 et 100 indiquant le paramètre de contrôle du volume. La valeur par défaut est 100.

Exemple

L'exemple suivant définit le volume initial sur 10, un réglage relativement bas.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
// Vous pouvez changer cette valeur de 0 à 100.
my_FLVPlybk.volume = 10;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Voir aussi

FLVPlayback.transform, FLVPlayback.volumeBar, FLVPlayback.volumeBarInterval, FLVPlayback.volumeUpdate

FLVPlayback.volumeBar

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.volumeBarInterval

Description

Propriété : objet MovieClip qui correspond à la commande de barre de volume. Pour plus d'informations sur l'utilisation des composants de l'interface utilisateur personnalisée Lecture de fichiers FLV pour les commandes de lecture, reportez-vous à « Application d'enveloppe aux composants particuliers de l'interface utilisateur personnalisée Lecture de fichiers FLV », à la page 544.

Exemple

L'exemple suivant utilise les propriétés backButton, forwardButton, playButton, pauseButton, stopButton et volumeBar pour associer des commandes d'interface utilisateur FLV personnalisées et individuelles à un composant FLVPlayback.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Dans l'inspecteur des composants, définissez le paramètre Skin (Enveloppe) sur None. Ajoutez les composants individuels suivants de l'interface utilisateur personnalisée FLV et donnez-leur les noms d'occurrence indiqués entre parenthèses : BackButton (**my_bkbttn**), ForwardButton (**my_fwdbttn**), PlayButton (**my_plybttn**), PauseButton (**my_pausbttn**), StopButton (**my_stopbttn**) et VolumeBar (**my_vBar**). Ajoutez ensuite les lignes de code suivantes au panneau Actions :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
  - composants d'interface utilisateur personnalisés FLV BackButton,
  ForwardButton, PlayButton, PauseButton, StopButton et VolumeBar dans la
  bibliothèque
*/
import mx.video.*;
my_FLVPlybk.backButton = my_bkbttn;
my_FLVPlybk.forwardButton = my_fwdbttn;
my_FLVPlybk.playButton = my_plybttn;
my_FLVPlybk.pauseButton = my_pausbttn;
my_FLVPlybk.stopButton = my_stopbttn;
my_FLVPlybk.volumeBar = my_vBar;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Voir aussi

FLVPlayback.volume, FLVPlayback.volumeBarInterval, FLVPlayback.volumeBarScrubTolerance, FLVPlayback.volumeUpdate

FLVPlayback.volumeBarInterval

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.volumeBarInterval

Description

Propriété : nombre indiquant la fréquence (en millisecondes) à laquelle la poignée de la barre de volume est activée lors de la modulation. La valeur par défaut est 250.

Exemple

L'exemple suivant définit la propriété `volumeBarInterval` sur 1 seconde (1 000 millisecondes) et crée un événement `volumeUpdate` qui affiche la durée de lecture et le volume à mesure que l'utilisateur fait glisser la poignée de la barre de volume. Les événements `volumeUpdate` se produisent approximativement à 1 seconde d'intervalle en raison de la définition de la propriété `volumeBarInterval`.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.volumeBarInterval = 1000;
var listenerObject:Object = new Object();
listenerObject.volumeUpdate = function(eventObject:Object) {
    trace("Playhead time is: " + my_FLVPlybk.playheadTime);
    trace("Volume is: " + my_FLVPlybk.volume);
};
my_FLVPlybk.addEventListener("volumeUpdate", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Voir aussi

`FLVPlayback.volume`, `FLVPlayback.volumeBar`, `FLVPlayback.volumeBarScrubTolerance`, `FLVPlayback.volumeUpdate`

FLVPlayback.volumeBarScrubTolerance

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.volumeBarScrubTolerance

Description

Propriété ; chiffre indiquant jusqu'où l'utilisateur peut déplacer la poignée de la barre de volume avant qu'une mise à jour se produise. La valeur est exprimée sous la forme d'un pourcentage. La valeur par défaut est 5.

Exemple

L'exemple suivant définit la propriété `volumeBarScrubTolerance` sur `20` et crée un événement `volumeUpdate` qui affiche le réglage du volume à mesure que l'utilisateur fait glisser la poignée de la barre de volume.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.volumeBarScrubTolerance = 20;
var listenerObject:Object = new Object();
listenerObject.volumeUpdate = function(eventObject:Object) {
    trace("Playhead time is: " + my_FLVPlybk.playheadTime);
    trace("Volume is: " + my_FLVPlybk.volume);
};
my_FLVPlybk.addEventListener("volumeUpdate", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Voir aussi

`FLVPlayback.volumeBar`, `FLVPlayback.volumeBarInterval`

FLVPlayback.volumeUpdate

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.volumeUpdate = function(eventObject:Object):Void {
  // Insertion du code de gestion de l'événement ici.
};
my_FLVplybk.addEventListener("volumeUpdate", listenerObject);
```

Description

Evénement : distribué lorsque le volume change à la suite du déplacement de la poignée de la commande volumeBar par l'utilisateur ou de la définition de la propriété `volume` dans ActionScript. L'objet événement a une propriété `volume`.

Exemple

L'exemple suivant affiche la valeur de la propriété `volume` dans le panneau Sortie correspondant à toutes les modifications apportées par l'utilisateur au volume.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
var listenerObject:Object = new Object();
listenerObject.volumeUpdate = function(eventObject:Object):Void {
  trace("Volume setting is: " + eventObject.volume);
};
my_FLVPlybk.addEventListener("volumeUpdate", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
```

Voir aussi

`FLVPlayback.addEventListener()`, `FLVPlayback.volume`, `FLVPlayback.volumeBar`

FLVPlayback.width

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.width

Description

Propriété : nombre indiquant la largeur de l'occurrence FLVPlayback sur la scène. Cette propriété affecte uniquement la largeur de l'occurrence FLVPlayback et n'inclut pas la largeur d'un fichier SWF d'enveloppe qui peut être chargé. Utilisez la propriété `width` de l'occurrence FLVPlayback, mais pas la propriété `MovieClip._width`, car la propriété `_width` risque de donner une autre valeur en cas de chargement d'un fichier SWF d'enveloppe.

Exemple

L'exemple suivant définit les propriétés `width` et `height`, ce qui déclenche un événement `resize`, car la valeur par défaut de la propriété `maintainAspectRatio` est définie sur `true`. Lorsque l'événement se produit, le gestionnaire d'événements affiche la largeur et la hauteur de l'occurrence FLVPlayback redimensionnée dans le panneau Sortie.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
// La propriété maintainAspectRatio (true par défaut) affecte
// les dimensions réelles.
my_FLVPlybk.width = 400;
my_FLVPlybk.height = 350;
var listenerObject:Object = new Object();
listenerObject.resize = function(eventObject:Object) {
  trace("Width is: " + eventObject.target.width + " Height is: " +
  eventObject.target.height);
};
my_FLVPlybk.addEventListener("resize", listenerObject);
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
```

Voir aussi

`FLVPlayback.height`, `FLVPlayback.setSize()`, `FLVPlayback.preferredHeight`, `FLVPlayback.preferredWidth`

FLVPlayback.x

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.x

Description

Propriété : nombre indiquant la coordonnée horizontale (emplacement) du lecteur vidéo. Cette propriété affecte uniquement l'emplacement horizontal de l'occurrence FLVPlayback et n'inclut pas l'emplacement d'un fichier SWF d'enveloppe qui peut modifier cet emplacement s'il est appliqué. Utilisez la propriété `x` de l'occurrence FLVPlayback, mais pas la propriété `MovieClip._x`, car la propriété `_x` risque de donner une autre valeur en cas de chargement d'un fichier SWF d'enveloppe.

Exemple

L'exemple suivant place l'occurrence FLVPlayback à 50 pixels de la gauche et 25 pixels du haut.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
my_FLVPlybk.x = 50;
my_FLVPlybk.y = 25;
```

Voir aussi

`FLVPlayback.y`

FLVPlayback.y

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

my_FLVPlybk.y

Description

Propriété : nombre indiquant la coordonnée verticale (emplacement) du lecteur vidéo. Cette propriété affecte uniquement l'emplacement vertical de l'occurrence FLVPlayback et n'inclut pas l'emplacement d'un fichier SWF d'enveloppe qui peut modifier cet emplacement s'il est appliqué. Utilisez la propriété `y` de l'occurrence FLVPlayback, mais pas la propriété `MovieClip._y`, car la propriété `_y` risque de donner une autre valeur en cas de chargement d'un fichier SWF d'enveloppe.

Exemple

L'exemple suivant place l'occurrence FLVPlayback à 25 pixels de la gauche et 50 pixels du haut.

Faites glisser un composant FLVPlayback sur la scène et nommez l'occurrence, **my_FLVPlybk**. Ajoutez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
/**
 Requiert :
  - Composant FLVPlayback sur la scène dont l'occurrence est nommée
  my_FLVPlybk
*/
import mx.video.*;
my_FLVPlybk.contentPath = "http://www.helpexamples.com/flash/video/
  water.flv";
my_FLVPlybk.x = 25;
my_FLVPlybk.y = 50;
```

Voir aussi

`FLVPlayback.x`

Classe VideoError

Héritage Error VideoError

Nom de classe ActionScript `mx.video.VideoError`

Les propriétés de la classe VideoError vous permettent de diagnostiquer les conditions d'erreur qui se produisent lorsque vous utilisez le composant FLVPlayback.

La classe mx.video.VideoError étend la classe Error.

REMARQUE	La classe VideoError est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Propriétés de la classe VideoError

Le tableau suivant répertorie les propriétés de la classe VideoError.

Propriété	Description
`VideoError.code`	Code d'erreur numérique.
`VideoError.DELETE_DEFAULT_PLAYER`	Nombre indiquant une tentative de suppression du lecteur vidéo par défaut.
`VideoError.DELETE_DEFAULT_PLAYER`	Nombre indiquant un point de repère non valide.
`VideoError.INVALID_CONTENT_PATH`	Nombre indiquant une valeur `contentPath` non valide.
`VideoError.INVALID_SEEK`	Nombre indiquant une recherche non valide.
`VideoError.INVALID_XML`	Nombre indiquant que du code XML non valide a été trouvé dans un fichier XML.
`VideoError.NO_BITRATE_MATCH`	Nombre qui indique l'impossibilité de trouver un fichier FLV par défaut correspondant à toute vitesse de transmission.
`VideoError.NO_CONNECTION`	Nombre indiquant que la méthode ne peut pas se connecter au serveur ou trouver le fichier FLV sur le serveur.
`VideoError.NO_CUE_POINT_MATCH`	Nombre indiquant qu'aucun point de repère correspondant n'a été trouvé.

VideoError.code

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`mx.video.VideoError.code`

Description

Code numérique qui identifie la condition d'erreur.

Exemple

L'exemple suivant affiche la condition d'erreur dans le panneau Sortie :

```
import mx.video.*;

try {
  ...
} catch (err:VideoError) {
  trace ("Error code is: " + err.code)
  ...
}
```

VideoError.DELETE_DEFAULT_PLAYER

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`mx.video.VideoError.DELETE_DEFAULT_PLAYER`

Description

Erreur 1007 qui se produit si vous appelez la méthode `FLVPlayback.closeVideoPlayer()` pour tenter de fermer le lecteur vidéo par défaut (numéro 0). Vous ne pouvez pas supprimer le lecteur vidéo par défaut.

Exemple

Le code suivant recherche le code d'erreur `DELETE_DEFAULT_PLAYER` :

```
import mx.video.*;

try {
   ...
} catch (err:VideoError) {

if (err.code == DELETE_DEFAULT_PLAYER) {
     ...
   }
}
```

Voir aussi

`FLVPlayback.activeVideoPlayerIndex`

VideoError.ILLEGAL_CUE_POINT

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`mx.video.VideoError.ILLEGAL_CUE_POINT`

Description

Erreur 1002 qui indique qu'un point de repère non valide a été trouvé.

Exemple

Le code suivant recherche le code d'erreur `ILLEGAL_CUE_POINT` :

```
try {
   ...
} catch (err:VideoError) {
   if (err.code == ILLEGAL_CUE_POINT) {
     ...
   }
}
```

Voir aussi

`FLVPlayback.cuePoint`

VideoError.INVALID_CONTENT_PATH

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`mx.video.VideoError.INVALID_CONTENT_PATH`

Description

Valeur de 1004 qui indique qu'une valeur `contentPath` non valide a été trouvée.

Exemple

Le code suivant recherche le code d'erreur `INVALID_CONTENT_PATH` :

```
import mx.video.*;

try {
   ...
} catch (err:VideoError) {

if (err.code == INVALID_CONTENT_PATH) {
     ...
   }
}
```

Voir aussi

`FLVPlayback.contentPath`

VideoError.INVALID_SEEK

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`mx.video.VideoError.INVALID_SEEK`

Description

Valeur de 1003 qui indique qu'une recherche non valide a été tentée.

Exemple

Le code suivant recherche le code d'erreur `INVALID_SEEK` :

```
import mx.video.*;

try {
   ...
} catch (err:VideoError) {

if (err.code == INVALID_SEEK) {
     ...
   }
}
```

Voir aussi

`FLVPlayback.seek()`

VideoError.INVALID_XML

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`mx.video.VideoError.INVALID_XML`

Description

Erreur 1005 indiquant que du code XML non valide a été trouvé. Cette erreur peut se produire lors du téléchargement ou de l'analyse d'un fichier SMIL. La propriété `VideoError.message` contient du texte qui décrit ce problème précis. Pour plus d'informations, voir « Utilisation d'un fichier SMIL », à la page 737.

Exemple

Le code suivant recherche le code d'erreur `INVALID_XML` :

```
import mx.video.*;

try {
   ...
} catch (err:VideoError) {

if (err.code == INVALID_XML) {
     ...
   }
}
```

Voir aussi

`FLVPlayback.contentPath`

VideoError.NO_BITRATE_MATCH

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

`mx.video.VideoError.NO_BITRATE_MATCH`

Description

Erreur 1006 indiquant qu'aucun fichier FLV par défaut répertorié ne correspond à la vitesse de transmission. Cette erreur se produit uniquement lors du téléchargement et de l'analyse d'un fichier SMIL. Pour plus d'informations, voir « Utilisation d'un fichier SMIL », à la page 737.

Exemple

Le code suivant recherche le code d'erreur `NO_BITRATE_MATCH` :

```
import mx.video.*;

try {
   ...
} catch (err:VideoError) {

if (err.code == NO_BITRATE_MATCH) {
     ...
   }
}
```

Voir aussi

`FLVPlayback.bitrate`

VideoError.NO_CONNECTION

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
mx.video.VideoError.NO_CONNECTION
```

Description

Erreur 1000 indiquant que la méthode ne peut pas se connecter au serveur ou trouver le fichier FLV sur le serveur.

Exemple

Le code suivant recherche le code d'erreur `NO_CONNECTION` :

```
import mx.video.*;

try {
   ...
} catch (err:VideoError) {

if (err.code == NO_CONNECTION) {
     ...
   }
}
```

VideoError.NO_CUE_POINT_MATCH

Disponibilité

Flash Player 8.

Edition

Flash Professional 8.

Utilisation

```
mx.video.VideoError.NO_CUE_POINT_MATCH
```

Description

Valeur de 1001 indiquant qu'aucun point de repère répondant aux critères n'a été trouvé.

Exemple

Le code suivant recherche le code d'erreur `NO_CUE_POINT_MATCH` :

```
import mx.video.*;

try {
   ...
} catch (err:VideoError) {

if (err.code == NO_CUE_POINT_MATCH) {
     ...
   }
}
```

Voir aussi

`FLVPlayback.findCuePoint()`

Classe VideoPlayer

Héritage MovieClip > Classe VideoPlayer

Nom de classe ActionScript `mx.video.VideoPlayer`

VideoPlayer étend la classe MovieClip et englobe un objet Video.

REMARQUE	La classe VideoPlayer est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

La classe FLVPlayback englobe la classe VideoPlayer, à ce sujet Adobe vous conseille vivement d'utiliser la classe FLVPlayback dans la plupart des cas. Aucune fonctionnalité de la classe VideoPlayer n'est accessible via la classe FLVPlayback.

La classe VideoPlayer est incluse ici, car elle vous permet de créer un lecteur vidéo avec un fichier SWF plus petit. Elle ne vous permet pas d'inclure des commandes d'enveloppe ou de lecture, et possède une API plus petite. Par exemple, vous ne pouvez pas rechercher de points de repère, même si des événements `cuePoint` se produisent.

Par ailleurs, la classe FLVPlayback crée automatiquement une interface avec la classe NCManager pour accéder aux fichiers FLV à diffusion en flux continu sur un serveur FMS, par exemple. Vous interagissez avec la classe NCManager lorsque vous définissez la propriété `contentPath` et lorsque vous transmettez une URL aux méthodes `play()` et `load()`. Cependant, si vous utilisez la classe VideoPlayer proprement dite, vous devez inclure l'instruction suivante dans votre code ActionScript afin de garantir l'inclusion de la classe NCManager :

```
_forceNCManager:mx.video.NCManager;
```

La classe NCManager possède également une classe d'interface, INCManager, qui vous permet de remplacer la classe NCManager par une classe personnalisée permettant de gérer les communications réseau. Le cas échéant, vous devez inclure l'instruction suivante, en remplaçant `NCManager` par le nom de la classe que vous avez indiqué :

```
mx.video.VideoPlayer.DEFAULT_INCMANAGER = "mx.video.NCManager";
```

Vous n'êtes pas tenu d'ajouter cette instruction si vous utilisez la classe NCManager par défaut.

REMARQUE	Vous pouvez également définir DEFAULT_INCMANAGER pour remplacer mx.video.NCManager par défaut par le composant FLVPlayback.

NCManager prend en charge un sous-ensemble de SMIL afin de gérer plusieurs flux continus pour plusieurs bandes passantes. Pour plus d'informations, voir « Utilisation d'un fichier SMIL », à la page 737.

Cette section récapitule la classe VideoPlayer. Vous pouvez trouver une documentation détaillée sur les méthodes, les propriétés et les événements de la classe VideoPlayer à l'adresse suivante www.adobe.com/go/videoplayer_fr.

Récapitulatif des méthodes de la classe VideoPlayer

Le tableau suivant répertorie les méthodes de la classe VideoPlayer.

Méthode	Description
`VideoPlayer.addEventListener()`	Crée un écouteur pour un événement spécifié.
`VideoPlayer.close()`	Ferme le flux vidéo et la connexion FMS.
`VideoPlayer.load()`	Charge le fichier FLV sans commencer la lecture. Le fichier FLV est mis en pause après le redimensionnement (si besoin).

Méthode	Description
`VideoPlayer.pause()`	Interrompt la lecture du flux vidéo en continu.
`VideoPlayer.play()`	Commence la lecture du flux vidéo en continu.
`VideoPlayer.removeEventListener()`	Supprime un écouteur d'événement.
`VideoPlayer.seek()`	Recherche une heure donnée (en secondes) dans le fichier, avec une précision décimale allant jusqu'aux millisecondes.
`VideoPlayer.setScale()`	Définit simultanément `scaleX` et `scaleY`.
`VideoPlayer.setSize()`	Définit simultanément les propriétés `width` et `height`.
`VideoPlayer.stop()`	Arrête la lecture du flux vidéo en continu.

Récapitulatif des propriétés de la classe VideoPlayer

La classe VideoPlayer possède des propriétés de classe et d'occurrence.

Propriétés de la classe

Les propriétés indiquées ci-dessous ne concernent que la classe VideoPlayer. Il s'agit de constantes en lecture seule qui s'appliquent à toutes les occurrences de la classe VideoPlayer.

Propriété	Valeur	Description
`VideoPlayer.BUFFERING`	`"buffering"`	Valeur possible de la propriété `state`. Indique l'état dans lequel le composant entre immédiatement après l'appel aux méthodes `play()` ou `load()`.
`VideoPlayer.CONNECTION_ERROR`	`"connectionError"`	Valeur possible de la propriété `state`. Indique une erreur de connexion.
`VideoPlayer.DEFAULT_INCMANAGER`	`"mx.video.NCManager"`	Nom de l'interface INCManager par défaut (mx.video.NCManager) ou implémentation personnalisée de cette interface.

Propriété	Valeur	Description
`VideoPlayer.DISCONNECTED`	`"disconnected"`	Valeur possible de la propriété `state`. Indique la déconnexion du flux continu du fichier FLV.
`VideoPlayer.LOADING`	`"loading"`	Valeur possible de la propriété `state`. Indique le chargement du fichier FLV.
`VideoPlayer.PAUSED`	`"paused"`	Valeur possible de la propriété `state`. Indique que le fichier FLV est en pause.
`VideoPlayer.PLAYING`	`"playing"`	Valeur possible de la propriété `state`. Indique que le fichier FLV est en cours de lecture.
`VideoPlayer.RESIZING`	`"resizing"`	Valeur possible de la propriété `state`. Indique que le fichier FLV est en cours de redimensionnement.
`VideoPlayer.REWINDING`	`"rewinding"`	Valeur possible de la propriété `state`. Indique que le fichier FLV est en cours de rembobinage.
`VideoPlayer.SEEKING`	`"seeking"`	Valeur possible de la propriété `state`. Indique que le fichier FLV est en cours de recherche.
`VideoPlayer.STOPPED`	`"stopped"`	Valeur possible de la propriété `state`. Indique que le fichier FLV est arrêté.
`VideoPlayer.version`	x.x.x.xx	Nombre correspondant au numéro de version du composant.

Propriétés des occurrences

Le tableau suivant répertorie les propriétés des occurrences de la classe VideoPlayer. Cet ensemble de propriétés s'applique à chaque occurrence d'une classe VideoPlayer.

Propriété	Description
`VideoPlayer.autoRewind`	Valeur booléenne qui, si elle est définie sur `true`, provoque le rembobinage du fichier FLV jusqu'à la première image lorsque la lecture s'arrête.
`VideoPlayer.autoSize`	Valeur booléenne qui, si elle est définie sur `true`, redimensionne automatiquement la vidéo aux dimensions sources.
`VideoPlayer.bufferTime`	Nombre de secondes à mettre en mémoire tampon avant de commencer la lecture d'un flux vidéo.
`VideoPlayer.bytesLoaded`	Nombre indiquant le degré de téléchargement en nombre d'octets pour un téléchargement HTTP. Lecture seule.
`VideoPlayer.bytesTotal`	Nombre total d'octets téléchargés pour un téléchargement HTTP. Lecture seule.
`VideoPlayer.connected`	Valeur booléenne indiquant si le flux du fichier FLV est connecté. Lecture seule.
`VideoPlayer.height`	Nombre indiquant la hauteur de la vidéo en pixels.
`VideoPlayer.idleTimeout`	urée (en millisecondes) avant qu'une connexion inactive au serveur FMS (en raison de l'interruption ou de l'arrêt de la lecture) soit arrêtée.
`VideoPlayer.isLive`	Valeur booléenne définie sur `true` si le flux vidéo est en direct. Ne s'applique pas à un téléchargement HTTP.
`VideoPlayer.isRTMP`	Valeur booléenne définie sur `true` si le fichier FLV est en cours de diffusion en flux continu à partir de FMS. Lecture seule.
`VideoPlayer.maintainAspectRatio`	Valeur booléenne qui, si elle est définie sur `true`, conserve les proportions de la vidéo.
`VideoPlayer.metadata`	Objet représentant un paquet d'informations de métadonnées reçu à partir d'un appel à la fonction de rappel `onMetaData()`, le cas échéant. Lecture seule.
`VideoPlayer.ncMgr`	Objet INCManager fournissant l'accès à une occurrence de la classe implémentant INCManager

Propriété	Description
`VideoPlayer.playheadTime`	Nombre représentant la durée de lecture (ou position de la tête de lecture) actuelle, mesurée en secondes, qui peut être une valeur décimale.
`VideoPlayer.playheadUpdateInterval`	Nombre représentant l'intervalle (en millisecondes) entre chaque événement `playheadUpdate`.
`VideoPlayer.progressInterval`	Nombre représentant l'intervalle (en millisecondes) entre chaque événement progress.
`VideoPlayer.scaleX`	Nombre indiquant le redimensionnement horizontal.
`VideoPlayer.scaleY`	Nombre indiquant le redimensionnement vertical.
`VideoPlayer.state`	Chaîne indiquant l'état du composant. Définie avec les méthodes `load()`, `play()`, `stop()`, `pause()` et `seek()`. Lecture seule.
`VideoPlayer.stateResponsive`	Valeur booléenne définie sur `true` si l'état est réactif, c'est-à-dire si les commandes peuvent être activées dans l'état actuel. Lecture seule.
`VideoPlayer.totalTime`	Chiffre représentant la durée de lecture totale pour la vidéo.
`VideoPlayer.transform`	Objet permettant d'accéder directement aux méthodes `Sound.setTransform()` et `Sound.getTransform()` afin de fournir un meilleur son.
`VideoPlayer.url`	Chaîne indiquant l'URL du flux continu chargé (ou en cours de chargement).
`VideoPlayer.videoHeight`	Nombre indiquant la hauteur du fichier FLV.
`VideoPlayer.videoWidth`	Nombre indiquant la largeur du fichier FLV.
`VideoPlayer.visible`	Valeur booléenne qui, si elle est définie sur `true`, rend le fichier FLV visible.
`VideoPlayer.volume`	Nombre compris entre 0 et 100 indiquant le réglage de contrôle du volume.
`VideoPlayer.width`	Nombre indiquant (en pourcentage) jusqu'où l'utilisateur peut déplacer la poignée de la barre de volume avant qu'une mise à jour se produise.
`VideoPlayer.x`	Chiffre indiquant la dimension horizontale du lecteur vidéo en pixels.
`VideoPlayer.y`	Chiffre indiquant la dimension verticale du lecteur vidéo en pixels.

Récapitulatif des événements de la classe VideoPlayer

Le tableau suivant répertorie les événements de la classe VideoPlayer.

Evénement	Description
`VideoPlayer.close`	Distribué lorsque le flux vidéo est fermé, via un timeout ou un appel à la méthode `close()`.
`VideoPlayer.complete`	Distribué lorsque la lecture se termine en atteignant la fin du fichier FLV.
`VideoPlayer.cuePoint`	Distribué lorsqu'un point de repère est atteint.
`VideoPlayer.metadataReceived`	Distribué la première fois que les métadonnées du fichier FLV sont atteintes.
`VideoPlayer.playheadUpdate`	Distribué toutes les 0,25 secondes lors de la lecture du fichier FLV.
`VideoPlayer.progress`	Distribué toutes les 0,25 secondes, en commençant lorsque la méthode `load()` est appelée et en se terminant lorsque tous les octets sont chargés ou en cas d'erreur réseau.
`VideoPlayer.ready`	Distribué lorsque le fichier FLV est chargé et prêt à être affiché.
`VideoPlayer.resize`	Distribué lors du redimensionnement de la vidéo.
`VideoPlayer.rewind`	Distribué lorsque l'emplacement de la tête de lecture recule suite à un appel à la méthode `seek()` ou à la fin du rembobinage automatique.
`VideoPlayer.stateChange`	Distribué lorsque l'état de lecture est modifié.

Utilisation d'un fichier SMIL

Afin de gérer plusieurs flux continus pour plusieurs bandes passantes, la classe VideoPlayer utilise une classe d'assistance (NCManager) qui prend en charge un sous-ensemble de SMIL. Le langage SMIL permet d'identifier l'emplacement du flux vidéo, la disposition (largeur et hauteur) du fichier FLV ainsi que les fichiers FLV source qui correspondent aux différentes bandes passantes. Il peut également être utilisé pour indiquer la vitesse de transmission et la durée du fichier FLV.

Utilisez le paramètre `contentPath` ou la propriété (ActionScript) `FLVPlayback.contentPath` pour spécifier l'emplacement d'un fichier SMIL. Pour plus d'informations, voir les sections « Chemin du contenu », à la page 529 et « FLVPlayback.contentPath » à la page 601.

L'exemple suivant affiche un fichier SMIL qui diffuse en continu plusieurs fichiers FLV à partir d'un serveur FMS à l'aide du protocole :

```
<smil>
    <head>
        <meta base="rtmp://myserver/mypgm/" >
        <layout>
            <root-layout width="240" height="180" >
        </layout>
    </head>
    <body>
        <switch>
            <video src="myvideo_mdm.flv" system-bitrate="56000"
  dur="3:00.1">
            <video src="myvideo_isdn.flv" system-bitrate="128000"
  dur="3:00.1">
            <ref src="myvideo_cable.flv" dur="3:00.1"/>
        </switch>
    </body>
</smil>
```

La balise `<head>` peut contenir les balises `<meta>` et `<layout>`. La balise `<meta>` ne prend en charge que l'attribut `base`, qui permet de spécifier l'URL de la vidéo en flux continu (RTMP à partir d'un serveur FMS).

La balise `<layout>` ne prend en charge que l'élément `root-layout` qui permet de définir les attributs `height` et `width` et donc de déterminer la taille de la fenêtre dans laquelle le fichier FLV est rendu. Ces attributs acceptent uniquement des valeurs en pixels et non des pourcentages.

Vous pouvez insérer dans le corps du fichier SMIL un lien vers un fichier source FLV ou, si vous diffusez en continu plusieurs fichiers pour plusieurs bandes passantes d'un serveur FMS (comme dans l'exemple précédent), vous pouvez utiliser la balise `<switch>` pour dresser la liste des fichiers source.

Les balises `video` et `ref` situées dans la balise `<switch>` sont synonymes : elles peuvent utiliser toutes deux l'attribut `src` pour spécifier des fichiers FLV. Par ailleurs, chacune peut utiliser les attributs `region`, `system-bitrate` et `dur` pour spécifier la région, la bande passante minimale requise et la durée du fichier FLV.

Dans la balise `<body>`, une seule occurrence des balises `<video>`, `<src>` ou `<switch>` est autorisée.

L'exemple suivant affiche le téléchargement progressif d'un seul fichier FLV qui n'utilise pas de détection de bande passante :

```
<smil>
     <head>
       <layout>
         <root-layout width="240" height="180" />
       </layout>
     </head>
     <body>
       <video src=""myvideo.flv" />
     </body>
</smil>
```

<smil>

Disponibilité

Flash Professional 8.

Utilisation

`<smil>`

...

`child tags`

...

</smil>

`Attributs`

Aucun.

Balises enfants

`<head>`, `<body>`

Balise parent

Aucun.

Description

Balise de niveau supérieur qui identifie un fichier SMIL.

Exemple

L'exemple suivant affiche un fichier SMIL spécifiant trois fichiers FLV :

```
<smil>
    <head>
        <meta base="rtmp://myserver/mypgm/" >
        <layout>
            <root-layout width="240" height="180" >
        </layout>
    </head>
    <body>
        <switch>
            <video src="myvideo_mdm.flv" system-bitrate="56000"
  dur="3:00.1">
            <video src="myvideo_isdn.flv" system-bitrate="128000"
  dur="3:00.1">
            <ref src="myvideo_cable.flv" dur="3:00.1"/>
        </switch>
    </body>
</smil>
```

<head>

Disponibilité

Flash Professional 8.

Utilisation

```
<head>
```

...

child tags

...

</head>

Attributs

Aucun.

Balises enfants

<meta>, <layout>

Balise parent

<smil>

Description

Prenant en charge les balises <meta> et <layout>, elle indique l'emplacement et la disposition par défaut (hauteur et largeur) des fichiers FLV source.

Exemple

L'exemple suivant définit la disposition racine sur 240 x 180 pixels :

```
<head>
  <meta base="rtmp://myserver/mypgm/" >
  <layout>
    <root-layout width="240" height="180" >
  </layout>
</head>
```

<meta>

Disponibilité

Flash Professional 8.

Utilisation

<meta/>

Attributs

base

Balises enfants

<layout>

Balise parent

Aucun.

Description

Contient l'attribut base qui spécifie l'emplacement (URL RTMP) des fichiers FLV sources.

Exemple

L'exemple suivant affiche une balise meta pour indiquer l'emplacement de base sur myserver :

```
<meta base="rtmp://myserver/mypgm/" >
```

<layout>

Disponibilité

Flash Professional 8.

Utilisation

```
<layout>
...
child tags
...
</layout>
```

Attributs

Aucun.

Balises enfants

<root-layout>

Balise parent

<meta>

Description

Indique la largeur et la hauteur du fichier FLV.

Exemple

L'exemple suivant spécifie la disposition sur 240 x 180 pixels :

```
<layout>
  <root-layout width="240" height="180" >
</layout>
```

<root-layout>

Disponibilité

Flash Professional 8.

Utilisation

<root-layout...*attributes*.../>

Attributs

Largeur, hauteur

Balises enfants

None.

Balise parent

`<layout>`

Description

Indique la largeur et la hauteur du fichier FLV.

Exemple

L'exemple suivant spécifie la disposition sur 240 x 180 pixels :

```
<root-layout width="240" height="180" >
```

‹body›

Disponibilité

Flash Professional 8.

Utilisation

```
<body>
...
child tags
...
</body>
```

`Attributs`

Aucun.

Balises enfants

`<video>`, `<ref>`, `<switch>`

Balise parent

`<smil>`

Description

Contient les balises `<video>`, `<ref>` et `<switch>` qui indiquent le nom du fichier FLV source, la bande passante minimale et la durée du fichier FLV. L'attribut `system-bitrate` est uniquement pris en charge pour utiliser la balise `<switch>`. Dans la balise `<body>`, une seule occurrence des balises `<video>`, `<ref>` ou `<switch>` est autorisée.

Exemple

L'exemple suivant spécifie trois fichiers FLV, deux utilisant la balise `video` et un utilisant la balise `ref` :

```
<body>
  <switch>
    <video src="myvideo_mdm.flv" system-bitrate="56000"  dur="3:00.1">
    <video src="myvideo_isdn.flv" system-bitrate="128000" dur="3:00.1">
    <ref src="myvideo_cable.flv" dur="3:00.1"/>
  </switch>
</body>
```

<video>

Disponibilité

Flash Professional 8.

Utilisation

```
<video...attributes.../>
```

Attributs

src, system-bitrate, dur

Balises enfants

None

Balise parent

<body>

Description

Synonyme de la balise <ref>. Elle prend en charge les attributs src et dur, qui spécifient le nom du fichier FLV source ainsi que sa durée. L'attribut dur prend en charge les formats horaires complet (00:03:00:01) et partiel (03:00:01).

Exemple

L'exemple suivant définit la source et la durée d'une vidéo :

```
<video src="myvideo_mdm.flv" dur="3:00.1">
```

<ref>

Disponibilité

Flash Professional 8.

Utilisation

```
<ref...attributes.../>
```

Attributs

src, system-bitrate, dur

Balises enfants

None

Balise parent

<body>

Description

Synonyme de la balise `<video>`. Elle prend en charge les attributs `src` et `dur`, qui spécifient le nom du fichier FLV source ainsi que sa durée. L'attribut `dur` prend en charge les formats horaires complet (00:03:00:01) et partiel (03:00:01).

Exemple

L'exemple suivant définit la source et la durée d'une vidéo :

```
<ref src="myvideo_cable.flv" dur="3:00.1"/>
```

<switch>

Disponibilité

Flash Professional 8.

Utilisation

```
<switch>
...
child tags
...
<switch/>
```

`Attributs`

Aucun.

Balises enfants

`<video>`, `<ref>`

Balise parent

`<body>`

Description

Utilisée avec les balises enfants `<video>` ou `<ref>` pour répertorier les fichiers FLV destinés à la diffusion de vidéo en continu via plusieurs bandes passantes. La balise `<switch>` prend en charge l'attribut `system-bitrate`, qui spécifie la bande passante minimale ainsi que les attributs `src` et `dur`.

Exemple

L'exemple suivant spécifie trois fichiers FLV, deux utilisant la balise `video` et un utilisant la balise `ref` :

```
<switch>
  <video src="myvideo_mdm.flv" system-bitrate="56000"  dur="3:00.1">
  <video src="myvideo_isdn.flv" system-bitrate="128000" dur="3:00.1">
  <ref src="myvideo_cable.flv" dur="3:00.1"/>
</switch>
```

CHAPITRE 23

Classe FocusManager

23

Vous pouvez utiliser la classe Focus Manager pour spécifier l'ordre dans lequel les composants reçoivent le focus lorsque l'utilisateur appuie sur la touche de tabulation pour parcourir une application. Elle permet également de définir un bouton dans votre document devant recevoir les entrées de clavier lorsque l'utilisateur appuie sur Entrée (Windows) ou Retour (Macintosh). Par exemple, lorsque l'utilisateur remplit un formulaire, il doit pouvoir passer d'un champ à l'autre en appuyant sur la touche de tabulation et envoyer le formulaire en appuyant sur Entrée (Windows) ou Retour (Macintosh).

Tous les composants prennent en charge la classe Focus Manager ; vous n'avez donc pas besoin d'écrire du code pour l'appeler.

REMARQUE

La classe FocusManager est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

CONSEIL

Focus Manager remplace le gestionnaire global `on(keyPress)`. Comme tous les composants implémentent Focus Manager, une application incluant des composants *et* utilisant le gestionnaire global `on(keyPress)` doit posséder un `tabIndex` pour tous les contrôles (y compris les composants *et* les clips) définis (voir « Utilisation de Focus Manager », à la page 746). Vous pouvez également ajouter un écouteur d'événement pour une touche spécifique, et la classe Focus Manager ne remplacera pas le gestionnaire d'événements correspondant (solution recommandée). Pour plus d'informations sur la création d'un écouteur d'événement pour une touche, reportez-vous à *Capture des pressions sur les touches* du guide *Formation à ActionScript 2.0* dans *Adobe Flash*.

Le gestionnaire de focus interagit avec le gestionnaire système, qui active et désactive les occurrences de FocusManager lorsque les fenêtres contextuelles s'ouvrent ou se ferment. Chaque fenêtre modale a une occurrence de FocusManager pour que ses composants définissent eux-mêmes la tabulation et empêchent ainsi l'utilisateur d'accéder aux composants des autres fenêtres avec la touche de tabulation.

Focus Manager reconnaît les groupes de boutons radio (ceux dont la propriété `RadioButton.groupName` est définie) et règle le focus sur l'occurrence du groupe dont la propriété `selected` est définie sur `true`. Lors d'un appui sur la touche de tabulation, il vérifie si l'objet suivant porte le même nom de groupe que l'objet en cours. Si tel est le cas, il déplace automatiquement le focus sur le prochain objet dont le nom de groupe est différent.
Les autres jeux de composants qui prennent en charge une propriété `groupName` peuvent également utiliser cette fonction.

Le gestionnaire de focus gère les changements de focus créés par les clics de souris.
Si l'utilisateur clique sur un composant, celui-ci reçoit le focus.

CONSEIL	Pour tester un script à l'aide de Focus Manager (Contrôle > Tester l'animation), choisissez Contrôle > Désactiver les raccourcis clavier en mode animation pour que Focus Manager fonctionne. Par défaut, la tabulation et les raccourcis clavier sont également utilisés par l'environnement de programmation. Ainsi, si vous utilisez le mode animation, l'utilisation des tabulations, de la touche Entrée et d'autres combinaisons de touches peut provoquer des résultats inattendus ou échouer. Ces fonctions doivent être testées dans Player hors de l'environnement de programmation.

Utilisation de Focus Manager

Le gestionnaire de focus n'affecte pas automatiquement le focus à un composant. Si vous souhaitez qu'un composant reçoive le focus lors du chargement d'une application, vous devez écrire un script qui appelle `FocusManager.setFocus()` sur un composant.

REMARQUE	Si vous appelez `FocusManager.setFocus()` pour attribuer le focus à un composant au chargement d'une application, le cercle du focus n'apparaît pas autour de ce composant. Le composant a le focus, mais l'indicateur n'est pas présent.

Pour créer une navigation de focus dans une application, définissez la propriété `tabIndex` sur tous les objets (y compris les boutons) qui doivent recevoir le focus. Lorsqu'un utilisateur appuie sur la touche de tabulation, Focus Manager recherche un objet activé pourvu d'une propriété `tabIndex` supérieure à la valeur actuelle de `tabIndex`. Dès qu'il a atteint la propriété `tabIndex` la plus élevée, il retombe à zéro. Dans l'exemple suivant, le focus est d'abord attribué à l'objet `comment` (probablement un composant TextArea), puis à l'objet `okButton` :

```
comment.tabIndex = 1;
okButton.tabIndex = 2;
```

Vous pouvez également utiliser le panneau Accessibilité pour affecter une valeur d'indexation.

Si aucune valeur d'indexation n'est définie sur la scène, Focus Manager utilise la *profondeur* (ordre d'empilement ou ordre *z*). La profondeur est configurée au départ par l'ordre dans lequel les composants sont placés sur la scène. Vous pouvez néanmoins utiliser les commandes Modifier > Réorganiser > Premier plan/Arrière-plan pour déterminer la profondeur finale.

Pour créer un bouton recevant le focus lorsque l'utilisateur appuie sur Entrée (Windows) ou sur Retour (Macintosh), définissez la propriété `FocusManager.defaultPushButton` sur le nom d'occurrence du bouton souhaité, comme indiqué dans le code suivant :

```
focusManager.defaultPushButton = okButton;
```

REMARQUE

Focus Manager tient compte de l'ordre de placement des objets sur la scène (leur ordre de profondeur) et non de leur position relative sur la scène. Cette particularité de traitement des tabulations le distingue de Flash Player.

Utilisation de Focus Manager pour autoriser la tabulation

Le gestionnaire de focus vous permet de créer un programme qui autorise l'utilisateur à parcourir les divers objets d'une application Flash en appuyant sur la touche de tabulation. (Les objets de ce programme de tabulation sont appelés *cibles de tabulation*.) Focus Manager examine les propriétés `tabEnabled` et `tabChildren` des parents des objets pour localiser ces derniers.

Un clip peut être un conteneur de cibles de tabulation, une cible de tabulation ou ni l'un ni l'autre :

Type de clip	tabEnabled	tabChildren
Conteneur de cibles de tabulation	`false`	`true`
Cible de tabulation	`true`	`false`
Aucun des deux	`false`	`false`

REMARQUE

Ce comportement diffère de celui de Flash Player, dans lequel la propriété `tabChildren` d'un conteneur peut être `undefined`.

Imaginons la situation suivante. La scène du scénario principal présente deux champs de texte (`txt1` et `txt2`) et un clip (`mc`) contenant un composant DataGrid (`grid1`) et un autre champ de texte (`txt3`). Vous pourriez dans ce cas utiliser le code suivant pour permettre aux utilisateurs d'appuyer sur la touche de tabulation et parcourir les objets dans l'ordre `txt1`, `txt2`, `grid1`, `txt3`.

REMARQUE

Les occurrences de FocusManager et de TextField sont activées par défaut.

```
// Focus Manager apprend que mc a des enfants.
// Ecrasement de mc.focusEnabled=true
mc.tabChildren=true;
mc.tabEnabled=false;
// Définition de la séquence de tabulation.
txt1.tabIndex = 1;
txt2.tabIndex = 2;
mc.grid1.tabIndex = 3;
mc.txt3.tabIndex = 4;

// Définition du focus initial sur txt1.
txt1.text = "focus";
focusManager.setFocus(txt1);
```

Si votre clip ne possède pas de méthode `onPress` ni `onRelease` ni de propriété `tabEnabled`, Focus Manager ne le verra que si vous définissez `focusEnabled` sur `true`. Les champs de saisie de texte sont toujours dans le programme de tabulation, sauf s'ils sont désactivés.

Lorsqu'une application Flash s'exécute dans un navigateur Web, elle n'a pas le focus tant que l'utilisateur n'a pas cliqué en un endroit quelconque de l'application. De même, dès que l'utilisateur y a cliqué, un appui sur la touche de tabulation peut faire passer le focus à l'extérieur de l'application Flash. Pour cantonner la tabulation aux objets de l'application Flash, dans le contrôle ActiveX de Flash Player 7, ajoutez le paramètre suivant à la balise HTML `<object>` :

```
<param name="SeamlessTabbing" value="false"/>
```

Création d'une application avec FocusManager

La procédure suivante crée un programme de focus dans une application Flash.

Pour créer un programme de focus :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser le composant TextInput du panneau Composants jusqu'à la scène.
3. Dans l'inspecteur Propriétés, donnez-lui le nom d'occurrence **comment**.
4. Faites glisser le composant Button du panneau Composants jusqu'à la scène.
5. Dans l'inspecteur Propriétés, affectez-lui le nom d'occurrence **okButton**, puis définissez le paramètre d'étiquette sur **OK**.
6. Dans l'image 1 du panneau Actions, saisissez le code suivant :

```
comment.tabIndex = 1;
okButton.tabIndex = 2;
focusManager.setFocus(comment);
function click(evt){
  trace(evt.type);
}
okButton.addEventListener("click", this);
```

7. Choisissez Contrôle > Tester l'animation.
8. Sélectionnez Contrôle > Désactiver les raccourcis clavier.

   Le code définit l'ordre de tabulation. Le champ de commentaire ne présente pas de cercle de focus initial mais a le focus initial, vous pouvez donc commencer à taper dans ce champ sans cliquer dedans au préalable. Vous devez également sélectionner l'option de menu Désactiver les raccourcis clavier pour que le focus fonctionne correctement en mode animation.

Personnalisation de Focus Manager

Vous pouvez changer la couleur du cercle du focus dans le thème Halo en modifiant la valeur du style `themeColor`, comme dans cet exemple :

```
_global.style.setStyle("themeColor", "haloBlue");
```

Focus Manager utilise une enveloppe FocusRect pour le tracé du focus. Cette enveloppe peut être remplacée ou modifiée, et les sous-classes peuvent se substituer à `UIComponent.drawFocus` pour le tracé des indicateurs de focus personnalisés.

Classe FocusManager (API)

Héritage MovieClip > Classe UIObject > Classe UIComponent > FocusManager

Nom de classe ActionScript mx.managers.FocusManager

Vous pouvez utiliser Focus Manager pour spécifier l'ordre dans lequel les composants reçoivent le focus lorsque l'utilisateur appuie sur la touche de tabulation pour parcourir une application. Cette classe permet également de définir un bouton dans votre document devant recevoir les entrées de clavier lorsque l'utilisateur appuie sur Entrée (Windows) ou Retour (Macintosh).

CONSEIL	Dans un fichier de classe qui hérite de la classe UIComponent, il n'est pas conseillé de faire référence à `_root.focusManager`. Chaque occurrence UIComponent hérite d'une méthode `getFocusManager()` qui renvoie une référence à l'occurrence FocusManager chargée du contrôle du programme de focus de ce composant.

Méthodes de la classe FocusManager

Le tableau suivant présente les méthodes de la classe FocusManager.

Méthode	Description
`FocusManager.getFocus()`	Renvoie une référence à l'objet ayant le focus.
`FocusManager.sendDefaultPushButtonEvent()`	Envoie un événement `click` aux objets d'écoute associés au bouton-poussoir par défaut.
`FocusManager.setFocus()`	Attribue le focus à l'objet spécifié.

Méthodes héritées de la classe UIObject

Le tableau suivant énumère les méthodes de la classe FocusManager héritées de la classe UIObject.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.

Méthode	Description
UIObject.getStyle()	Obtient la propriété de style de l'objet ou de la déclaration de style.
UIObject.invalidate()	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
UIObject.move()	Déplace l'objet à l'emplacement demandé.
UIObject.redraw()	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
UIObject.setSize()	Redimensionne l'objet à la taille demandée.
UIObject.setSkin()	Définit une enveloppe dans l'objet.
UIObject.setStyle()	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant énumère les méthodes de la classe FocusManager héritées de la classe UIComponent.

Méthode	Description
UIComponent.getFocus()	Renvoie une référence à l'objet ayant le focus.
UIComponent.setFocus()	Attribue le focus à l'occurrence de composant.

Propriétés de la classe FocusManager

Le tableau suivant présente les propriétés de la classe FocusManager.

Propriété	Description
FocusManager.defaultPushButton	Objet qui reçoit un événement `click` lorsque l'utilisateur appuie sur la touche Retour ou Entrée.
FocusManager.defaultPushButtonEnabled	Indique si le traitement clavier du bouton-poussoir par défaut est activé (`true`) ou désactivé (`false`). La valeur par défaut est `true`.
FocusManager.enabled	Indique si le traitement tabulation est activé (`true`) ou désactivé (`false`). La valeur par défaut est `true`.
FocusManager.nextTabIndex	Valeur suivante de la propriété `tabIndex`.

Propriétés héritées de la classe UIObject

Le tableau suivant énumère les propriétés de la classe FocusManager héritées de la classe UIObject.

Propriété	Description
`UIObject.bottom`	Position du bord inférieur de l'objet par rapport au bord inférieur de son parent. Lecture seule.
`UIObject.height`	Hauteur de l'objet, en pixels. Lecture seule.
`UIObject.left`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.right`	Position du bord droit de l'objet par rapport au bord droit de son parent. Lecture seule.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Position du bord supérieur de l'objet par rapport à son parent. Lecture seule.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Largeur de l'objet, en pixels. Lecture seule.
`UIObject.x`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.y`	Bord supérieur de l'objet, en pixels. Lecture seule.

Propriétés héritées de la classe UIComponent

Le tableau suivant énumère les propriétés de la classe FocusManager héritées de la classe UIComponent.

Propriété	Description
`UIComponent.enabled`	Indique si le composant peut recevoir le focus et la saisie.
`UIComponent.tabIndex`	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Evénement de la classe FocusManager

La classe FocusManager ne possède pas d'événement exclusif.

Evénements hérités de la classe UIObject

Le tableau suivant énumère les événements de la classe FocusManager hérités de la classe UIObject.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant énumère les événements de la classe FocusManager hérités de la classe UIComponent.

Evénement	Description
`UIComponent.focusIn`	Diffusé lorsqu'un objet reçoit le focus.
`UIComponent.focusOut`	Diffusé lorsqu'un objet perd le focus.
`UIComponent.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIComponent.keyUp`	Diffusé lorsqu'une touche est relâchée.

FocusManager.defaultPushButton

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004 ou Flash MX Professional 2004.

Utilisation

```
focusManager.defaultPushButton
```

Description

Propriété qui désigne le bouton-poussoir par défaut d'une application. Lorsque l'utilisateur appuie sur la touche Entrée (Windows) ou Retour (Macintosh), les écouteurs du bouton poussoir par défaut reçoivent un événement `click`. La valeur par défaut est `undefined` et les données de cette propriété sont de type objet.

Le gestionnaire de focus utilise la déclaration de style de mise en valeur de la classe SimpleButton pour signaler le bouton-poussoir par défaut visuellement.

La valeur de la propriété `defaultPushButton` est toujours le bouton qui a le focus.
La définition de la propriété `defaultPushButton` n'attribue pas le focus initial au bouton poussoir par défaut. Si une application comprend plusieurs boutons, celui qui a le focus reçoit l'événement `click` lorsque la touche Entrée ou Retour est enfoncée. Si le focus se trouve sur un autre composant lorsque la touche Entrée ou Retour est enfoncée, la valeur d'origine de la propriété `defaultPushButton` est rétablie.

Exemple

Le code suivant définit le bouton poussoir par défaut sur l'occurrence `OKButton` :

```
focusManager.defaultPushButton = OKButton;
```

Voir aussi

`FocusManager.defaultPushButtonEnabled`,
`FocusManager.sendDefaultPushButtonEvent()`

FocusManager.defaultPushButtonEnabled

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
focusManager.defaultPushButtonEnabled
```

Description

Propriété : valeur booléenne qui détermine si le traitement clavier du bouton poussoir par défaut est activé (`true`) ou non (`false`). La définition de `defaultPushButtonEnabled` sur `false` permet à un composant de recevoir la touche Retour ou Entrée et de la traiter en interne. Vous devez réactiver le traitement du bouton poussoir par défaut en observant la méthode `onKillFocus()` du composant (voir le gestionnaire `MovieClip.onKillFocus` dans le *Guide de référence du langage ActionScript 2.0*) ou l'événement `focusOut`. La valeur par défaut est `true`.

Cette propriété est réservée aux développeurs expérimentés.

Exemple

Le code suivant désactive le traitement du bouton-poussoir par défaut :

```
focusManager.defaultPushButtonEnabled = false;
```

FocusManager.enabled

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
focusManager.enabled
```

Description

Propriété : valeur booléenne qui détermine si le traitement tabulation est activé (`true`) ou non (`false`) pour un groupe spécifique d'objets de focus. Une autre fenêtre contextuelle pourrait avoir son propre FocusManager, par exemple. La définition de `enabled` sur `false` permet à un composant de recevoir les touches de traitement tabulation et de les traiter en interne.
Vous devez réactiver le traitement du bouton poussoir par défaut en observant la méthode `onKillFocus()` du composant (voir le gestionnaire `MovieClip.onKillFocus` dans le *Guide de référence du langage ActionScript 2.0*) ou l'événement `focusOut`. La valeur par défaut est `true`.

Exemple

Le code suivant désactive la tabulation :

```
focusManager.enabled = false;
```

FocusManager.getFocus()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004 ou Flash MX Professional 2004.

Utilisation

```
focusManager.getFocus()
```

Paramètres

Aucun.

Valeur renvoyée

Une référence à l'objet ayant le focus.

Description

Méthode qui renvoie une référence à l'objet ayant le focus.

Exemple

Le code suivant définit le focus sur `myOKButton` si l'objet ayant le focus est `myInputText` :

```
if (focusManager.getFocus() == myInputText)
{
  focusManager.setFocus(myOKButton);
}
```

Voir aussi

`FocusManager.setFocus()`

FocusManager.nextTabIndex

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
FocusManager.nextTabIndex
```

Description

Propriété, prochain numéro d'index de tabulation disponible. Utilisez cette propriété pour définir de façon dynamique la propriété `tabIndex` d'un objet.

Exemple

Le code suivant donne à l'occurrence `mycheckbox` la prochaine valeur `tabIndex` la plus élevée :

```
mycheckbox.tabIndex = focusManager.nextTabIndex;
```

Voir aussi

`UIComponent.tabIndex`

FocusManager.sendDefaultPushButtonEvent()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004 ou Flash MX Professional 2004.

Utilisation

```
focusManager.sendDefaultPushButtonEvent()
```

Paramètres

Aucun.

Valeur renvoyée

Aucune.

Description

Méthode : envoie un événement `click` aux objets écouteurs associés au bouton poussoir par défaut. Appliquez cette méthode pour envoyer un événement `click` par programmation.

Exemple

Le code suivant déclenche l'événement `click` du bouton poussoir par défaut et renseigne les champs de nom d'utilisateur et de mot de passe lorsqu'un utilisateur sélectionne l'occurrence CheckBox `chb` (la case à cocher s'appellerait « Automatic Login ») :

```
name_txt.tabIndex = 1;
password_txt.tabIndex = 2;
chb.tabIndex = 3;
submit_ib.tabIndex = 4;

focusManager.defaultPushButton = submit_ib;

chbObj = new Object();
chbObj.click = function(o){
  if (chb.selected == true){
    name_txt.text = "Jody";
    password_txt.text = "foobar";
    focusManager.sendDefaultPushButtonEvent();
  } else {
    name_txt.text = "";
    password_txt.text = "";
  }
}
chb.addEventListener("click", chbObj);

submitObj = new Object();
submitObj.click = function(o){
  if (password_txt.text != "foobar"){
    trace("error on submit");
  } else {
    trace("Yeah! sendDefaultPushButtonEvent worked!");
  }
}
submit_ib.addEventListener("click", submitObj);
```

Voir aussi

`FocusManager.defaultPushButton`

FocusManager.setFocus()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004 ou Flash MX Professional 2004.

Utilisation

```
focusManager.setFocus(object)
```

Paramètres

object Référence à l'objet qui reçoit le focus.

Valeur renvoyée

Aucune.

Description

Méthode qui attribue le focus à l'objet spécifié. Si l'objet auquel vous souhaitez donner le focus n'est pas dans le scénario principal, utilisez le code suivant :

```
_root.focusManager.setFocus(object);
```

Exemple

Le code suivant définit le focus sur `myOKButton` :

```
focusManager.setFocus(myOKButton);
```

Voir aussi

`FocusManager.getFocus()`

CHAPITRE 24

Classe Form

24

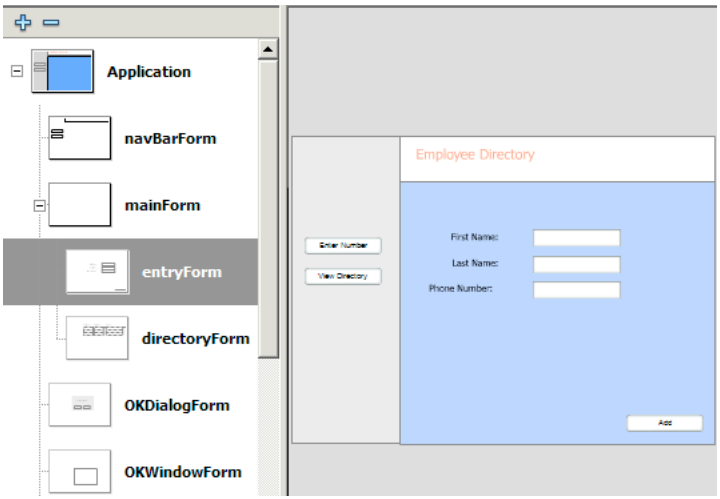
La classe Form définit le comportement à l'exécution des formulaires créés dans le panneau Contour de l'écran d'Adobe Flash CS3 Professional. La classe Form est une sous-classe de la classe Screen. Pour une présentation générale de l'utilisation des écrans, reportez-vous à *Utilisation des écrans* dans *Utilisation de Flash*.

REMARQUE

La fonction Ecrans est délaissée dans Flash CS3. Vous pouvez ouvrir et modifier des fichiers FLA composés d'écrans qui ont été créés à partir des précédentes versions de Flash, mais vous ne pouvez plus créer ce type de document. Par ailleurs, notez que la classe Form est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript2.0 dans ses paramètres de publication.

Utilisation de la classe Form

Les formulaires fonctionnent comme des conteneurs pour les objets graphiques (des éléments d'interface utilisateur dans une application, par exemple) et comme des états d'application. Vous pouvez utiliser le panneau Contour de l'écran pour visualiser les différents états d'une application que vous créez, où chaque formulaire est un état d'application différent. L'illustration suivante montre le panneau Contour de l'écran pour un exemple d'application conçu à l'aide de formulaires.



Vue du Contour de l'écran d'un exemple d'application de formulaires

Cette illustration montre le contour d'un exemple d'application appelé « Employee Directory », qui comprend plusieurs formulaires. Le formulaire nommé « entryForm » (sélectionné dans l'illustration ci-dessus) contient plusieurs objets d'interface utilisateur, y compris des champs de saisie, des étiquettes et un bouton-poussoir. Le développeur peut facilement présenter ce formulaire à l'utilisateur en faisant basculer sa visibilité (à l'aide de la propriété `Form.visible`) et celle des autres formulaires simultanément.

Vous pouvez également associer des comportements et des commandes aux formulaires à l'aide du panneau Comportements. Pour plus d'informations sur le sujet, reportez-vous à *Création de commandes et de transitions pour les écrans à l'aide des comportements* dans *Utilisation de Flash*.

Etant donné que la classe Form étend la classe Loader, vous pouvez aisément charger du contenu externe (un fichier SWF ou JPEG) dans un formulaire. Par exemple, le contenu d'un formulaire peut être un fichier SWF distinct, contenant lui-même des formulaires. De cette façon, vous pouvez modulariser vos applications de formulaires, ce qui facilite leur maintenance et réduit leur temps de téléchargement. Pour plus d'informations, voir « Chargement de contenu externe dans des écrans », à la page 1120.

Paramètres de la classe Form

Dans l'inspecteur Propriétés ou des composants, vous pouvez définir les paramètres de programmation suivants pour chaque occurrence de Form :

autoload indique si le contenu désigné par le paramètre contentPath doit être chargé automatiquement (`true`) ou attendre l'appel à la méthode `Loader.load()` (`false`). La valeur par défaut est `true`.

contentPath désigne le contenu du formulaire. Il peut s'agir de l'identificateur de liaison d'un clip ou de l'URL absolue ou relative d'un fichier SWF ou JPEG à charger dans la diapositive. Par défaut, le contenu chargé est coupé pour s'adapter à la diapositive.

visible précise si le formulaire apparaît (`true`) ou non (`false`) lors de son premier chargement.

Form, classe

Héritage MovieClip > Classe UIObject > Classe UIComponent > View > Composant Loader > Classe Screen > Form

Nom de classe ActionScript mx.screens.Form

La classe Form définit le comportement à l'exécution des formulaires créés dans le panneau Contour de l'écran de Flash.

Méthodes de la classe Form

Le tableau suivant présente les méthodes de la classe Form.

Méthode	Description
`Form.getChildForm()`	Renvoie le formulaire enfant à un index spécifié.

Méthodes héritées de la classe UIObject

Le tableau suivant énumère les méthodes de la classe Form héritées de la classe UIObject. Pour appeler ces méthodes à partir de l'objet Form, utilisez la syntaxe *`formInstance.methodName`*.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.

Méthode	Description
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
`UIObject.setSize()`	Redimensionne l'objet à la taille demandée.
`UIObject.setSkin()`	Définit une enveloppe dans l'objet.
`UIObject.setStyle()`	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant énumère les méthodes de la classe Form héritées de la classe UIComponent. Pour appeler ces méthodes à partir de l'objet Form, utilisez la syntaxe *`formInstance.methodName`*.

Méthode	Description
`UIComponent.getFocus()`	Renvoie une référence à l'objet ayant le focus.
`UIComponent.setFocus()`	Attribue le focus à l'occurrence de composant.

Méthodes héritées de la classe Loader

Le tableau suivant énumère les méthodes de la classe Form héritées de la classe Loader. Pour appeler ces méthodes à partir de l'objet Form, utilisez la syntaxe *`formInstance.methodName`*.

Méthode	Description
`Loader.load()`	Charge le contenu spécifié par la propriété `contentPath`.

Méthodes héritées de la classe Screen

Le tableau suivant énumère les méthodes de la classe Form héritées de la classe Screen. Pour appeler ces méthodes à partir de l'objet Form, utilisez la syntaxe *formInstance.methodName*.

Méthode	Description
Screen.getChildScreen()	Renvoie l'écran enfant de cet écran à un index spécifique.

Propriétés de la classe Form

Le tableau suivant présente les propriétés réservées à la classe Form.

Propriété	Description
Form.currentFocusedForm	(Lecture seule) ; renvoie le formulaire qui contient le focus global actuel.
Form.indexInParentForm	(Lecture seule) ; renvoie l'index (basé sur zéro) de ce formulaire dans la liste des sous-formulaires de son parent.
Form.numChildForms	(Lecture seule) ; renvoie le nombre de formulaires enfants que ce formulaire contient.
Form.parentIsForm	(Lecture seule) ; indique si l'objet parent de ce formulaire est également un formulaire.
Form.parentForm	(Lecture seule) ; référence au formulaire parent.
Form.rootForm	(Lecture seule) ; renvoie la racine de l'arborescence ou sous-arborescence du formulaire qui contient le formulaire.
Form.visible	Spécifie si le formulaire est visible lorsque son formulaire, sa diapositive, son clip ou son fichier SWF parent l'est.

Propriétés héritées de la classe UIObject

Le tableau suivant énumère les propriétés de la classe Form héritées de la classe UIObject. Pour accéder à ces propriétés à partir de l'objet Form, utilisez la syntaxe *formInstance.propertyName*.

Propriété	Description
UIObject.bottom	Position du bord inférieur de l'objet par rapport au bord inférieur de son parent. Lecture seule.
UIObject.height	Hauteur de l'objet, en pixels. Lecture seule.
UIObject.left	Bord gauche de l'objet, en pixels. Lecture seule.

Propriété	Description
`UIObject.right`	Position du bord droit de l'objet par rapport au bord droit de son parent. Lecture seule.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Position du bord supérieur de l'objet par rapport à son parent. Lecture seule.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Largeur de l'objet, en pixels. Lecture seule.
`UIObject.x`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.y`	Bord supérieur de l'objet, en pixels. Lecture seule.

Propriétés héritées de la classe UIComponent

Le tableau suivant énumère les propriétés de la classe Form héritées de la classe UIComponent. Pour accéder à ces propriétés à partir de l'objet Form, utilisez la syntaxe *`formInstance.propertyName`*.

Propriété	Description
`UIComponent.enabled`	Indique si le composant peut recevoir le focus et la saisie.
`UIComponent.tabIndex`	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Propriétés héritées de la classe Loader

Le tableau suivant énumère les propriétés de la classe Form héritées de la classe Loader. Pour accéder à ces propriétés à partir de l'objet Form, utilisez la syntaxe *`formInstance.propertyName`*.

Propriété	Description
`Loader.autoLoad`	Valeur booléenne indiquant si le contenu se charge automatiquement (`true`) ou si vous devez appeler `Loader.load()` (`false`).
`Loader.bytesLoaded`	(lecture seule), indique le nombre d'octets ayant été chargés.
`Loader.bytesTotal`	(lecture seule), indique le nombre d'octets total du contenu.

Propriété	Description
`Loader.content`	Référence au contenu du composant Loader. Cette propriété est en lecture seule.
`Loader.contentPath`	Chaîne indiquant l'URL du contenu devant être chargé.
`Loader.percentLoaded`	Nombre indiquant le pourcentage de contenu chargé. Cette propriété est en lecture seule.
`Loader.scaleContent`	Valeur booléenne indiquant si le contenu est dimensionné pour s'ajuster au chargeur (`true`) ou si le chargeur est dimensionné pour s'ajuster au contenu (`false`).

Propriétés héritées de la classe Screen

Le tableau suivant répertorie les propriétés de la classe Form héritées de la classe Screen. Pour accéder à ces propriétés à partir de l'objet Form, utilisez la syntaxe *`formInstance.propertyName`*.

Propriété	Description
`Screen.currentFocusedScreen`	(Lecture seule), renvoie l'écran qui contient le focus global actuel.
`Screen.indexInParent`	Lecture seule ; renvoie l'index de l'écran (dont la numérotation commence à zéro) dans la liste des écrans enfants de son écran parent.
`Screen.numChildScreens`	(Lecture seule), renvoie le nombre d'écrans enfant contenus dans l'écran.
`Screen.parentIsScreen`	Lecture seule : renvoie une valeur booléenne (`true` ou `false`) indiquant si l'objet parent de l'écran est lui-même un écran.
`Screen.rootScreen`	(Lecture seule), renvoie l'écran racine de l'arborescence ou sous-arborescence qui contient l'écran.

Evénements de la classe Form

La classe Form ne possède pas d'événement exclusif.

Evénements hérités de la classe UIObject

Le tableau suivant énumère les événements de la classe Form hérités de la classe UIObject.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant énumère les événements de la classe Form hérités de la classe UIComponent.

Evénement	Description
`UIComponent.focusIn`	Diffusé lorsqu'un objet reçoit le focus.
`UIComponent.focusOut`	Diffusé lorsqu'un objet perd le focus.
`UIComponent.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIComponent.keyUp`	Diffusé lorsqu'une touche est relâchée.

Evénements hérités de la classe Loader

Le tableau suivant énumère les événements de la classe Form hérités de la classe Loader.

Evénement	Description
`Loader.complete`	Déclenché à la fin du chargement du contenu.
`Loader.progress`	Déclenché pendant le chargement du contenu.

Evénements hérités de la classe Screen

Le tableau suivant énumère les événements de la classe Form hérités de la classe Screen.

Evénement	Description
`Screen.allTransitionsInDone`	Diffusé lorsque toutes les transitions « en entrée » appliquées à un écran sont terminées.
`Screen.allTransitionsOutDone`	Diffusé lorsque toutes les transitions « en sortie » appliquées à un écran sont terminées.
`Screen.mouseDown`	Diffusé lorsque l'utilisateur clique sur un objet (forme ou clip) appartenant directement à l'écran.
`Screen.mouseDownSomewhere`	Diffusé lorsque l'utilisateur clique sur un endroit quelconque de la scène, mais pas nécessairement sur un objet appartenant à l'écran.
`Screen.mouseMove`	Diffusé lorsque le pointeur de la souris bouge alors qu'il se trouve sur un écran.
`Screen.mouseOut`	Diffusé lorsque le pointeur de la souris passe de l'intérieur à l'extérieur de l'écran.
`Screen.mouseOver`	Diffusé lorsque le pointeur de la souris se déplace de l'extérieur vers l'intérieur de l'écran.
`Screen.mouseUp`	Diffusé lorsque l'utilisateur relâche le bouton de la souris au-dessus d'un objet (forme ou clip) appartenant directement à l'écran.
`Screen.mouseUpSomewhere`	Diffusé lorsque l'utilisateur relâche le bouton de la souris quelque part sur la scène, mais pas nécessairement sur un objet appartenant à l'écran.

Form.currentFocusedForm

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
mx.screens.Form.currentFocusedForm
```

Description

Propriété (lecture seule) qui renvoie l'objet Form contenant le focus global actuel. Le focus actuel peut être sur le formulaire lui-même ou sur un clip, un objet texte ou un composant dans le formulaire. Peut être `null` s'il n'y a aucun focus actuellement.

Exemple

Le code suivant, associé à un bouton (masqué), affiche le nom du formulaire ayant le focus actuel.

```
trace("The form with the current focus is: " +
  mx.screens.Form.currentFocusedForm);
```

Form.getChildForm()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
myForm.getChildForm(childIndex)
```

Paramètres

`childIndex` Nombre indiquant l'index (basé sur zéro) du formulaire enfant à renvoyer.

Renvoie

Un objet Form.

Description

Méthode : renvoie le formulaire enfant de *myForm* dont l'index est *childIndex*.

Exemple

L'exemple suivant affiche dans le panneau Sortie les noms de tous les objets Form enfants qui appartiennent à l'objet racine Form nommé `Application`.

```
for (var i:Number = 0; i < _root.application.numChildForms; i++) {
    var childForm:mx.screens.Form = _root.application.getChildForm(i);
    trace(childForm._name);
}
```

Voir aussi

`Form.numChildForms`

Form.indexInParentForm

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myForm`.indexInParentForm`

Description

Propriété (lecture seule) qui contient l'index (basé sur zéro) de *myForm* dans la liste des formulaires enfants de son parent. Si l'objet parent de *myForm* est un écran et non un formulaire (par exemple, une diapositive), alors `indexInParentForm` est toujours 0.

Exemple

```
var myIndex:Number = myForm.indexInParent;
if (myForm == myForm._parent.getChildForm(myIndex)) {
  trace("I'm where I should be");
}
```

Voir aussi

`Form.getChildForm()`

Form.numChildForms

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myForm.numChildForms

Description

Propriété (lecture seule) : nombre de formulaires enfants contenus dans *myForm* provenant directement de la classe mx.screens.Form. Cette propriété ne comprend aucune diapositive contenue dans *myForm*, seulement des formulaires.

REMARQUE	Lors de l'utilisation d'une classe ActionScript 2.0 personnalisée qui étend mx.screens.Form, le formulaire n'est pas pris en compte dans la propriété numChildForms.

Exemple

Le code suivant se répète sur tous les formulaires enfants contenus dans myForm et affiche leurs noms dans le panneau Sortie.

```
var howManyKids:Number = myForm.numChildForms;
for(i=0; i<howManyKids; i++) {
  var childForm = myForm.getChildForm(i);
  trace(childForm._name);
}
```

Voir aussi

Form.getChildForm()

Form.parentIsForm

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myForm.parentIsForm

Description

Propriété (lecture seule) : renvoie une valeur booléenne indiquant si l'objet parent du formulaire spécifié est également un formulaire (`true`) ou non (`false`). Si cette propriété est `false`, *myForm* est à la racine de sa hiérarchie de formulaires.

Exemple

```
if (myForm.parentIsForm) {
  trace("I have "+myForm._parent.numChildScreens+" sibling screens");
} else {
  trace("I am the root form and have no siblings");
}
```

Form.parentForm

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myForm.parentForm

Description

Propriété (lecture seule) : référence au formulaire parent.

Exemple

L'exemple de code suivant réside dans un écran nommé `myForm` correspondant à un enfant de l'écran `form1` par défaut créé lorsque vous avez sélectionné Application du formulaire Flash dans la boîte de dialogue Nouveau document.

```
onClipEvent(keyDown){
  var parentForm:mx.screens.Form = this.parentForm;
  trace(parentForm);
}
// résultat : _level0.application.form1
```

Form.rootForm

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myForm.rootForm

Description

Propriété (lecture seule) qui renvoie le formulaire au sommet de la hiérarchie de formulaires contenant *myForm*. Si *myForm* est contenu dans un objet qui n'est pas un formulaire (c'est-à-dire une diapositive), cette propriété renvoie alors *myForm*.

Exemple

Dans l'exemple suivant, une référence au formulaire racine de myForm est placée dans une variable nommée root. Si la valeur affectée à root fait référence à myForm, alors myForm est au sommet de son arborescence de formulaires.

```
var root:mx.screens.Form = myForm.rootForm;
if(root == myForm) {
  trace("myForm is the top form in its tree");
}
```

Form.visible

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myForm.visible

Description

Propriété : détermine si *myForm* est visible lorsque son formulaire, sa diapositive, son clip ou son fichier SWF parent l'est. Vous pouvez également définir cette propriété à l'aide de l'inspecteur Propriétés dans l'environnement de programmation de Flash.

Lorsque cette propriété est définie sur `true`, *myForm* reçoit un événement `reveal`. Lorsqu'elle est définie sur `false`, *myForm* reçoit un événement `hide`. Vous pouvez associer aux formulaires des transitions qui s'exécutent lorsqu'un formulaire reçoit l'un de ces événements. Pour plus d'informations sur l'ajout de transitions et de commandes aux écrans, reportez-vous à *Création de commandes et de transitions pour les écrans à l'aide des comportements* dans *Utilisation de Flash*.

Exemple

Dans une image du scénario, le code suivant définit la propriété `visible` du formulaire qui contient le bouton sur `false`.

```
btnOk.addEventListener("click", btnOkClick);
function btnOkClick(eventObj:Object):Void {
  eventObj.target._parent.visible = false;
}
```

CHAPITRE 25

Interface Iterator

25

Nom de classe ActionScript mx.utils.Iterator

L'interface Iterator permet de parcourir les objets d'une collection.

REMARQUE

L'interface Iterator est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Méthodes de l'interface Iterator

Le tableau suivant présente les méthodes de l'interface Iterator.

Méthode	Description
`Iterator.hasNext()`	Indique si l'itérateur contient plus d'éléments.
`Iterator.next()`	Renvoie l'élément suivant de l'itération.

Iterator.hasNext()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
iterator.hasNext()
```

Renvoie

Valeur booléenne indiquant si l'itérateur contient (`true`) ou non (`false`) d'autres occurrences.

Description

Méthode qui indique si l'itérateur contient d'autres occurrences. Cette méthode est généralement utilisée dans une instruction `while` lorsque l'on effectue une boucle par l'intermédiaire d'un itérateur.

Exemple

L'exemple suivant utilise la méthode `hasNext()` pour contrôler une boucle effectuée via l'itérateur des éléments d'une collection :

```
on (click) {
  var myColl:mx.utils.Collection;
  myColl = _parent.thisShelf.MyCompactDisks;
  var itr:mx.utils.Iterator = myColl.getIterator();
  while (itr.hasNext()) {
    var cd:CompactDisk = CompactDisk(itr.next());
    var title:String = cd.Title;
    var artist:String = cd.Artist;
    trace("Title: "+title+" Artist: "+artist);
  }
}
```

Iterator.next()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`iterator``.next()`

Renvoie

Objet correspondant au prochain élément de l'itérateur.

Description

Méthode qui renvoie une occurrence du prochain élément de l'itérateur. Vous devez attribuer le type approprié à cette occurrence.

Exemple

L'exemple suivant utilise la méthode `next()` pour accéder à l'élément suivant d'une collection :

```
on (click) {
  var myColl:mx.utils.Collection;
  myColl = _parent.thisShelf.MyCompactDisks;
  var itr:mx.utils.Iterator = myColl.getIterator();
  while (itr.hasNext()) {
    var cd:CompactDisk = CompactDisk(itr.next());
    var title:String = cd.Title;
    var artist:String = cd.Artist;
    trace("Title: "+title+" Artist: "+artist);
  }
}
```

CHAPITRE 26

Composant Label

26

Un composant Label est une ligne unique de texte. Vous pouvez demander le formatage HTML d'une étiquette. Vous pouvez également contrôler son alignement et sa taille. Les étiquettes n'ont pas de bordures, ne peuvent pas recevoir le focus et ne diffusent pas d'événements.

REMARQUE

Un composant Label est pris en charge pour ActionScript 2.0 et ActionScript 3.0. Ce document présente le composant de la version 2. Si vous utilisez celui de la version 3, reportez-vous à « Utilisation du composant Label » dans *Utilisation des composants ActionScript 3.0*.

L'aperçu en direct des occurrences Label reflète les modifications apportées aux paramètres dans l'inspecteur Propriétés ou des composants pendant la programmation. Le composant Label n'ayant pas de bordures, la définition de son paramètre de texte constitue le seul moyen de visualiser son aperçu en direct. Le paramètre autoSize n'est pas pris en charge dans l'aperçu en direct.

Lorsque vous ajoutez le composant Label à une application, vous pouvez utiliser le panneau Accessibilité pour le rendre accessible aux lecteurs d'écran. Commencez par ajouter la ligne de code suivante pour activer l'accessibilité :

```
mx.accessibility.LabelAccImpl.enableAccessibility();
```

Quel que soit le nombre d'occurrences d'un composant, l'activation de son accessibilité ne se fait qu'une fois. Pour plus d'informations, reportez-vous à *Création de contenu accessible* dans *Utilisation de Flash*.

Utilisation du composant Label

Utilisez des composants Label afin de créer des étiquettes de texte pour les autres composants de formulaire, tels que l'étiquette « Nom : ». placée à gauche d'un champ TextInput où l'on saisirait un nom d'utilisateur. Il est préférable d'utiliser un composant Label au lieu d'un champ de texte ordinaire car vous pouvez vous servir des styles pour maintenir une cohésion au niveau de la présentation.

Si vous souhaitez faire pivoter un composant Label, vous devez intégrer les polices.
Voir « Utilisation des styles avec le composant Label », à la page 784.

Paramètres du composant Label

Dans l'inspecteur des propriétés ou l'inspecteur de composants (Fenêtre > Inspecteur de composants), vous pouvez définir les paramètres de création suivants pour chaque occurrence Label :

autoSize indique le dimensionnement et l'alignement de l'étiquette pour contenir le texte. La valeur par défaut est `none`. Ce paramètre peut prendre l'une des quatre valeurs suivantes :

- `none` indique que l'étiquette n'est pas redimensionnée ni alignée pour contenir le texte.
- `left` indique que le bas et le côté droit de l'étiquette sont redimensionnés pour contenir le texte. Le haut et le côté gauche ne sont pas redimensionnés.
- `center` indique que les côtés gauche et droit de l'étiquette sont redimensionnés pour contenir le texte. Le centre horizontal de l'étiquette reste ancré à sa position d'origine.
- `right` indique que le bas et le côté gauche de l'étiquette sont redimensionnés pour contenir le texte. Le haut et le côté droit ne sont pas redimensionnés.

REMARQUE

La propriété `autoSize` du composant Label est différente de la propriété `autoSize` de l'objet TextField ActionScript intégré.

html indique si l'étiquette est mise au format HTML (`true`) ou non (`false`). Si ce paramètre est défini sur `true`, une étiquette ne peut pas être mise en forme avec des styles, mais vous pouvez mettre le texte au format HTML à l'aide de la balise `font`. La valeur par défaut est `false`.

text indique le texte de l'étiquette ; la valeur par défaut est `Label`.

Dans l'inspecteur des composants, vous pouvez définir les paramètres supplémentaires suivants pour chaque occurrence du composant Label (Fenêtre > Inspecteur de composants) :

visible est une valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`). La valeur par défaut est `true`.

REMARQUE

Les propriétés `minHeight` et `minWidth` sont utilisées par des routines de dimensionnement internes. Elles sont définies dans UIObject et remplacées par différents composants selon vos besoins. Ces propriétés peuvent être utilisées pour créer un gestionnaire de dispositions personnalisées. Dans le cas contraire, la définition de ces propriétés dans l'inspecteur de composants n'a aucun effet visible.

Vous pouvez rédiger des instructions ActionScript qui définissent d'autres options pour les occurrences d'étiquette à l'aide des méthodes, des propriétés et des événements. Pour plus d'informations, voir « Classe Label », à la page 785.

Création d'une application avec le composant Label

La procédure suivante décrit l'ajout d'un composant Label à une application au cours de la programmation. Dans cet exemple, l'étiquette se trouve à côté d'une liste déroulante contenant des dates, dans une application de panier d'achat.

Pour créer une application avec le composant Label :

1. Vérifiez que vos paramètres de publication sont bien définis sur ActionScript 2.0.
2. Faites glisser un composant Label du panneau Composants vers la scène.
3. Dans l'inspecteur de composants, entrez le paramètre d'étiquette **Expiration Date**.

Pour créer une occurrence du composant Label à l'aide d'ActionScript :

1. Vérifiez que vos paramètres de publication sont bien définis sur ActionScript 2.0.
2. Faites glisser le composant Label du panneau Composants jusqu'à la bibliothèque du document actuel.

   Cette opération permet d'ajouter le composant à la bibliothèque, mais elle ne permet pas de le rendre visible dans l'application.
3. Sélectionnez la première image dans le scénario principal, ouvrez le panneau Actions et indiquez le code suivant :

   ```
   this.createClassObject(mx.controls.Label, "my_label", 1);
   my_label.text = "Hello World";
   ```

   Ce script utilise la méthode `UIObject.createClassObject()` pour créer l'occurrence du composant Label.
4. Choisissez Contrôle > Tester l'animation.

Personnalisation du composant Label

Vous pouvez orienter un composant Label horizontalement et verticalement pendant la création et à l'exécution. Lors de la programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. Vous pouvez également définir le paramètre de programmation autoSize ; la définition de ce paramètre ne change pas le cadre de sélection dans l'aperçu en direct, mais l'étiquette est redimensionnée. Pour plus d'informations, voir « Paramètres du composant Label », à la page 782. Lors de l'exécution, utilisez la méthode `setSize()` (voir `UIObject.setSize()` ou `Label.autoSize`).

Utilisation des styles avec le composant Label

Vous pouvez définir des propriétés de style afin de modifier l'aspect d'une occurrence d'étiquette. Tout le texte contenu dans une occurrence de composant Label doit partager le même style. Par exemple, vous ne pouvez pas définir le style `color` sur `"blue"` pour un mot d'une étiquette et le définir sur `"red"` pour un autre mot de la même étiquette.

Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, reportez-vous à « Utilisation de styles pour personnaliser la couleur et le texte des composants » dans *Utilisation des composants ActionScript 2.0*.

Les composants Label prennent en charge les styles suivants :

Style	Thème	Description
`color`	Les deux	Couleur du texte. La valeur par défaut est 0x0B333C pour le thème Halo et vide pour le thème Sample.
`disabledColor`	Les deux	Couleur du texte lorsque le composant est désactivé. La couleur par défaut est 0x848384 (gris foncé).
`embedFonts`	Les deux	Valeur booléenne qui indique si la police spécifiée dans `fontFamily` est de type intégré. Ce style doit être défini sur `true` si `fontFamily` fait référence à une police intégrée. Dans le cas contraire, la police intégrée n'est pas utilisée. Si ce style est défini sur `true` et que `fontFamily` ne fait pas référence à une police intégrée, aucun texte n'est affiché. La valeur par défaut est `false`.
`fontFamily`	Les deux	Nom de la police du texte. La valeur par défaut est `"_sans"`.
`fontSize`	Les deux	Taille de la police, en points. La valeur par défaut est 10.

Style	Thème	Description
`fontStyle`	Les deux	Style de police : « `normal` » ou « `italic` ». La valeur par défaut est `"normal"`.
`fontWeight`	Les deux	Epaisseur de la police : `"none"` ou `"bold"`. La valeur par défaut est `"none"`. Tous les composants peuvent également accepter la valeur `"normal"` au lieu de `"none"` pendant un appel à la méthode `setStyle()`, mais les appels suivants à `getStyle()` renvoient `"none"`.
`textAlign`	Les deux	Alignement du texte : `"left"`, `"right"` ou `"center"`. La valeur par défaut est `"left"`.
`textDecoration`	Les deux	Décoration du texte : `"none"` ou `"underline"`. La valeur par défaut est `"none"`.

Utilisation des enveloppes avec le composant Label

Le composant Label ne présente aucun élément visuel susceptible de recevoir des enveloppes.

Classe Label

Héritage MovieClip > Classe UIObject > Label

Nom de classe ActionScript mx.controls.Label

Lors de l'exécution, les propriétés de la classe Label permettent de spécifier du texte pour l'étiquette, d'indiquer si ce texte peut être converti au format HTML et de spécifier si l'étiquette sera automatiquement ajustée au texte.

La définition d'une propriété de la classe Label avec ActionScript annule le paramètre du même nom défini dans l'inspecteur Propriétés ou des composants.

Si vous accédez aux valeurs des propriétés du composant Label, vérifiez qu'il est totalement chargé avant d'accéder à la propriété souhaitée. Considérons l'exemple suivant :

```
var listenerObject:Object = new Object();
listenerObject.load = function(){
  trace(label.width);
};
label.addEventListener("load", listenerObject);
```

Toutes les classes de composants ont une propriété `version` qui est une propriété de classe. Les propriétés de classe ne sont disponibles que dans la classe elle-même. La propriété `version` renvoie une chaîne qui indique la version du composant. Pour accéder à cette propriété, utilisez le code suivant :

```
trace(mx.controls.Label.version);
```

REMARQUE

Le code `trace(myLabelInstance.version);` renvoie `undefined`.

Méthodes de la classe Label

La classe Label ne présente pas de méthodes exclusives.

Méthodes héritées de la classe UIObject

Le tableau suivant énumère les méthodes de la classe Label héritées de la classe UIObject. Pour appeler ces méthodes à partir de l'objet Label, utilisez le formulaire *`labelInstance.methodName`*.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
`UIObject.setSize()`	Redimensionne l'objet à la taille demandée.
`UIObject.setSkin()`	Définit une enveloppe dans l'objet.
`UIObject.setStyle()`	Définit la propriété de style sur l'objet ou la déclaration de style.

Propriétés de la classe Label

Le tableau suivant présente les propriétés de la classe Label.

Propriété	Description
`Label.autoSize`	Chaîne qui indique la manière dont une étiquette doit être dimensionnée et alignée pour s'adapter à la valeur de sa propriété `text`. Quatre valeurs sont possibles : `"none"`, `"left"`, `"center"` et `"right"`. La valeur par défaut est `"none"`.
`Label.html`	Valeur booléenne qui indique si une étiquette peut être mise au format HTML (`true`) ou non (`false`).
`Label.text`	Texte de l'étiquette.

Propriétés héritées de la classe UIObject

Le tableau suivant énumère les propriétés de la classe Label héritées de la classe UIObject. Pour accéder à ces propriétés, utilisez le formulaire *`labelInstance.propertyName`*.

Propriété	Description
`UIObject.bottom`	Position du bord inférieur de l'objet par rapport au bord inférieur de son parent. Lecture seule.
`UIObject.height`	Hauteur de l'objet, en pixels. Lecture seule.
`UIObject.left`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.right`	Position du bord droit de l'objet par rapport au bord droit de son parent. Lecture seule.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Position du bord supérieur de l'objet par rapport à son parent. Lecture seule.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Largeur de l'objet, en pixels. Lecture seule.
`UIObject.x`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.y`	Bord supérieur de l'objet, en pixels. Lecture seule.

Evénements de la classe Label

La classe Label ne présente pas d'événement exclusif.

Evénements hérités de la classe UIObject

Le tableau suivant énumère les événements de la classe Label hérités de la classe UIObject.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

Label.autoSize

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`labelInstance``.autoSize`

Description

Propriété : chaîne indiquant la manière dont une étiquette est dimensionnée et alignée pour s'adapter à la valeur de sa propriété `text`. Quatre valeurs sont possibles : `"none"`, `"left"`, `"center"` et `"right"`. La valeur par défaut est `"none"`.

- `none` L'étiquette n'est pas redimensionnée ni alignée pour contenir le texte.
- `left` Le bas et le côté droit de l'étiquette sont redimensionnés pour contenir le texte. Le haut et le côté gauche ne sont pas redimensionnés.

- `right` Le bas et le côté gauche de l'étiquette sont redimensionnés pour contenir le texte. Le centre horizontal de l'étiquette reste ancré à sa position d'origine.
- `right` Le bas et le côté gauche de l'étiquette sont redimensionnés pour contenir le texte. Le haut et le côté droit ne sont pas redimensionnés.

REMARQUE

La propriété `autoSize` du composant Label est différente de la propriété `autoSize` de l'objet TextField ActionScript intégré.

Exemple

Dans l'exemple suivant, l'occurrence du composant Label `my_label` redimensionne le bas et le côté gauche de l'étiquette pour contenir tout le texte :

```
my_label.text = "A really long label with Label.autoSize set";
my_label.autoSize = "right";
```

Label.html

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`labelInstance``.html`

Description

Propriété : valeur booléenne indiquant si l'étiquette peut être mise au format HTML (`true`) ou non (`false`). La valeur par défaut est `false`. Les composants Label dont la propriété `html` est définie sur `true` ne peuvent pas être mis en forme avec des styles.

Pour extraire le texte brut d'un texte au format HTML, définissez la propriété `HTML` sur `false`, puis accédez à la propriété `text`. Cette opération ayant pour effet de supprimer le formatage HTML, il est conseillé de copier le texte de l'étiquette dans un composant Label ou TextArea hors écran avant de l'exécuter.

Exemple

L'exemple suivant définit la propriété `html` sur `true` pour que l'étiquette puisse être mise au format HTML. La propriété `text` est ensuite définie sur une chaîne contenant une mise en forme HTML.

```
my_label.html = true;
my_label.text = "The <b>Royal</b> Nonesuch";
my_label.autoSize = "right";
```

Le mot « Royal » est affiché en caractères gras.

Label.text

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`labelInstance``.text`

Description

Propriété, texte d'une étiquette. La valeur par défaut est `"Label"`.

Exemple

Le code suivant définit la propriété `text` de l'occurrence Label `my_label` et envoie la valeur vers le panneau Sortie :

```
my_label.text = "The Royal Nonesuch";
trace(my_label.text);
```

CHAPITRE 27

Composant List

27

Le composant List est une zone de liste défilante à sélection unique ou multiple. Les listes peuvent également contenir des graphiques et d'autres composants. L'ajout des éléments affichés dans la zone de liste s'effectue via la boîte de dialogue Valeurs qui s'ouvre lorsque vous cliquez dans les champs de paramètres des étiquettes ou des données. Vous pouvez également utiliser les méthodes `List.addItem()` et `List.addItemAt()` pour ajouter des éléments à la liste.

REMARQUE

Un composant List est pris en charge pour ActionScript 2.0 et ActionScript 3.0. Ce document présente le composant de la version 2. Si vous utilisez celui de la version 3, reportez-vous à « Utilisation du composant List » dans *Utilisation des composants ActionScript 3.0*.

Le composant List utilise un index basé sur zéro, où l'élément possédant l'index 0 est le premier affiché. Lorsque vous ajoutez, supprimez ou remplacez les éléments d'une liste au moyen des méthodes et des propriétés de la classe List, il peut s'avérer nécessaire d'indiquer leur index.

La liste reçoit le focus lorsque l'utilisateur clique sur son entrée ou appuie sur la touche de tabulation pour y accéder. Les touches suivantes permettent de la contrôler :

Touche	Description
Touches alphanumériques	Passe à l'élément suivant dont `Key.getAscii()` est le premier caractère de l'étiquette.
Ctrl	Bouton bascule autorisant plusieurs sélections et désélections non contiguës.
Flèche vers le bas	Déplace la sélection d'un élément vers le bas.
Origine	Déplace la sélection jusqu'au sommet de la liste.
Pg. Suiv.	Déplace la sélection sur la page suivante.
Pg. Préc.	Déplace la sélection sur la page précédente.
Maj	Permet une sélection contiguë.
Flèche vers le haut	Déplace la sélection d'un élément vers le haut.

REMARQUE

La taille de page utilisée par les touches Page précédente et Page suivante correspond au nombre d'éléments contenus dans l'affichage, moins un. Par exemple, le passage à la page suivante dans une liste déroulante à dix lignes affiche les éléments 0-9, 9-18, 18-27, etc., avec un élément commun par page.

Pour plus d'informations sur le contrôle du focus, reportez-vous à « Classe FocusManager », à la page 745 ou à « Création de la navigation personnalisée du focus » dans *Utilisation des composants ActionScript 2.0*.

L'aperçu en direct de chaque occurrence de List sur la scène reflète les modifications apportées aux paramètres dans l'inspecteur Propriétés ou des composants au cours de la programmation.

Lorsque vous ajoutez le composant List à une application, vous pouvez utiliser le panneau Accessibilité pour le rendre accessible aux lecteurs d'écran. Commencez par ajouter la ligne de code suivante pour activer l'accessibilité :

```
mx.accessibility.ListAccImpl.enableAccessibility();
```

Quel que soit le nombre d'occurrences d'un composant, l'activation de son accessibilité ne se fait qu'une fois. Pour plus d'informations, reportez-vous à *Création de contenu accessible* dans *Utilisation de Flash*.

Utilisation du composant List

Vous pouvez définir une liste dans laquelle les utilisateurs pourront effectuer un ou plusieurs choix. Par exemple, un utilisateur qui visite un site de commerce électronique doit pouvoir choisir l'article à acheter. Imaginons que 30 articles lui soient proposés. Il fait défiler la liste et en choisit un en cliquant sur son entrée.

Vous pouvez également concevoir une liste qui affiche des lignes de clips personnalisés pour mieux renseigner les utilisateurs. Par exemple, dans une application de courrier électronique, chaque boîte de réception peut être un composant List et chaque ligne peut être accompagnée d'icônes indiquant la priorité et l'état des messages.

Fonctionnement du composant List

Lors du développement d'une application avec le composant List, ou d'un composant étendant la classe List, il est important de bien comprendre la conception de la classe List. Voici quelques hypothèses et éléments requis utilisés par Adobe lors du développement de la classe List :

- Petite, rapide et simple.

  Ne jamais compliquer les choses plus que nécessaire. Voilà la première consigne à appliquer. La majorité des éléments requis énumérés ci-dessous respectent cette directive.
- Hauteurs de lignes des listes uniformes.

  Toutes les lignes doivent être de même hauteur. Celle-ci peut être définie à la programmation ou au moment de l'exécution.
- Les listes peuvent comporter des milliers d'enregistrements.
- Les listes ne mesurent pas le texte.

  Cette restriction a le plus de ramifications potentielles. Les listes pouvant comporter des milliers d'enregistrements et l'un d'eux pouvant contenir une chaîne exceptionnellement longue, elles ne doivent pas s'ajuster à la chaîne la plus grande ni ajouter une barre de défilement horizontal en mode « auto ». De même, mesurer des milliers de chaînes se révèlerait fastidieux. Le compromis existe avec la propriété `maxHPosition` qui, lorsque `vScrollPolicy` est défini sur « `on` », donne à la liste un espace tampon supplémentaire pour le défilement.

  Si vous savez que vous serez amené à traiter de longues chaînes, définissez `hScrollPolicy` sur « `on` », puis ajoutez une valeur `maxHPosition` de 200 pixels à votre composant List ou Tree. L'utilisateur est alors plus ou moins assuré de pouvoir faire défiler les éléments pour tout voir. Le composant DataGrid prend toutefois en charge la valeur « `auto` » pour `hScrollPolicy`, car il mesure les colonnes (de même largeur par élément) et non le texte.

Le fait que les listes ne mesurent pas le texte explique également l'uniformité des hauteurs de lignes. Le dimensionnement individuel de chaque ligne en fonction du texte demanderait un très grand nombre de mesures. Par exemple, pour afficher avec précision les barres de défilement d'une liste de lignes aux hauteurs non uniformes, il vous faudrait d'abord mesurer chaque ligne.

- Les listes s'exécutent d'autant moins bien que le nombre de lignes visibles est important.

  Si elles peuvent contenir 5000 enregistrements, elles ne peuvent pas tous les afficher simultanément. Plus le nombre de lignes visibles sur la scène est important (défini par la propriété `rowCount`), plus la liste doit travailler au rendu. Lorsque cela est possible, limiter le nombre de lignes visibles est la meilleure solution.

- Les listes ne sont pas des tableaux.

  Par exemple, les composants DataGrid, qui étendent la classe List, sont utilisés pour afficher de nombreux enregistrements. Ils ne sont pas conçus pour afficher la totalité des informations, mais pour en présenter suffisamment pour que l'utilisateur puisse ensuite les développer. L'affichage des messages dans Microsoft Outlook en est un bon exemple. Il n'est pas nécessaire de lire la totalité du message dans la grille. La lecture s'avérerait très difficile et le client de messagerie aurait bien du mal à s'exécuter. Outlook présente suffisamment d'informations pour que l'utilisateur en explore la totalité s'il est intéressé.

Paramètres du composant List

Dans l'inspecteur Propriétés ou des composants, vous pouvez définir les paramètres de programmation suivants pour chaque composant List :

data est un tableau de valeurs qui alimente la liste en données. La valeur par défaut est `[]` (tableau vide). Il n'existe pas de propriété équivalente lors de l'exécution.

labels est un tableau des valeurs de texte qui remplissent les étiquettes de la liste. La valeur par défaut est `[]` (tableau vide). Il n'existe pas de propriété équivalente lors de l'exécution.

multipleSelection est une valeur booléenne qui indique si plusieurs valeurs peuvent être sélectionnées (`true`) ou non (`false`). La valeur par défaut est `false`.

rowHeight indique la hauteur de chaque ligne, en pixels. La valeur par défaut est 20. La définition d'une police ne change rien à la hauteur de la ligne.

Vous pouvez rédiger du code ActionScript afin de définir d'autres options pour les occurrences de liste à l'aide des méthodes, des propriétés et des événements. Pour plus d'informations, voir « Classe List », à la page 801.

Création d'une application avec le composant List

La procédure suivante décrit l'ajout d'un composant List à une application au cours de la programmation. Dans cet exemple, la liste est un exemple comportant trois éléments.

Pour ajouter un composant List simple à une application :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser un composant List du panneau Composants sur la scène.
3. Sélectionnez l'outil Transformer librement et redimensionnez le composant pour l'adapter à votre application.
4. Dans l'inspecteur Propriétés, procédez comme suit :
   - Entrez le nom d'occurrence **my_list**.
   - Entrez **Item1**, **Item2** et **Item3** pour le paramètre des étiquettes.
   - Entrez **item1.html**, **item2.html**, **item3.html** pour le paramètre des données.
5. Choisissez Contrôle > Tester l'animation pour visualiser la liste et ses éléments.
6. Retournez dans l'environnement de programmation, insérez un nouveau calque et nommez-le **actions**.
7. Dans l'image 1 du calque Actions, ajoutez le code ActionScript suivant.

```
my_list.change = function(evt:Object) {
  getURL(evt.target.selectedItem.data, "_blank");
};
my_list.addEventListener("change", my_list);
```

Pour alimenter une occurrence List avec un fournisseur de données :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser un composant List du panneau Composants sur la scène.
3. Sélectionnez l'outil Transformer librement et redimensionnez le composant pour l'adapter à votre application.
4. Dans l'inspecteur Propriétés, entrez le nom d'occurrence **my_list**.
5. Sélectionnez l'image 1 du scénario et, dans le panneau Actions, entrez le code suivant :

   ```
   my_list.dataProvider = myDP;
   ```

   Si vous avez défini un fournisseur de données nommé `myDP`, la liste se remplit de données. (Pour plus d'informations sur les fournisseurs de données, reportez-vous à `List.dataProvider`.)
6. Choisissez Contrôle > Tester l'animation pour visualiser la liste et ses éléments.

Pour contrôler une occurrence de clip à l'aide d'un composant List :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser un composant List du panneau Composants sur la scène.
3. Sélectionnez l'outil Transformer librement et redimensionnez le composant pour l'adapter à votre application.
4. Dans l'inspecteur Propriétés, entrez le nom d'occurrence **my_list**.
5. Créez un clip sur la scène et nommez-le **my_mc**.
6. Ouvrez le clip en mode édition et ajoutez une animation.
7. Insérez un nouveau calque et nommez-le **actions**.
8. Dans l'image 1 du calque Actions, ajoutez le code ActionScript suivant.

```
my_list.addItem({label:"play", data:"play"});
my_list.addItem({label:"stop", data:"stop"});
var listHandler:Object = new Object();
listHandler.change = function(evt:Object) {
  switch (evt.target.selectedItem.data) {
  case "play" :
    my_mc.play();
    break;
  case "stop" :
    my_mc.stop();
    break;
  default :
    trace("unhandled event: "+evt.target.selectedItem.data);
    break;
  }
};
my_list.addEventListener("change", listHandler);
```

9. Choisissez Contrôle > Tester l'animation pour utiliser la liste afin d'arrêter et de lire l'occurrence de clip `my_mc`.

Pour créer une occurrence du composant List à l'aide d'ActionScript :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser le composant List du panneau Composants jusqu'à la bibliothèque.

   Cette opération permet d'ajouter le composant à la bibliothèque, mais elle ne permet pas de le rendre visible dans l'application.

3. Sélectionnez la première image dans le scénario principal, ouvrez le panneau Actions et indiquez le code suivant :

```
this.createClassObject(mx.controls.List, "my_list", 1);
my_list.addItem({label:"One", data:dt1});
my_list.addItem({label:"Two", data:dt2});
```

   Ce script utilise la méthode `UIObject.createClassObject()` pour créer l'occurrence List.

4. Choisissez Contrôle > Tester l'animation.

Personnalisation du composant List

Vous pouvez orienter un composant List horizontalement et verticalement pendant la création et l'exécution. Lors de la programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez la méthode `List.setSize()` (voir `UIObject.setSize()`).

Lorsqu'une liste est redimensionnée, ses lignes diminuent horizontalement, ce qui tronque leur texte. Verticalement, la liste ajoute ou supprime le nombre de lignes approprié. Les barres de défilement se placent automatiquement. Pour plus d'informations sur les barres de défilement, reportez-vous à « Composant ScrollPane », à la page 1141.

Utilisation des styles avec le composant List

Vous pouvez définir des propriétés de style afin de modifier l'aspect d'un composant List.

Les composants List utilisent les styles suivants :

Style	Thème	Description
`themeColor`	Halo	Couleur de base d'un composant. `"haloGreen"`, `"haloBlue"` et `"haloOrange"` sont des valeurs possibles. La valeur par défaut est `"haloGreen"`.
`alternatingRowColors`	Les deux	Spécifie des couleurs alternées pour les lignes. La valeur peut être un tableau de couleurs, 0xFF00FF, 0xCC6699 et 0x996699, par exemple. Contrairement aux styles de couleur à valeur simple, `alternatingRowColors` n'accepte pas les noms de couleur. Les valeurs doivent être des codes de couleur numériques. Par défaut, ce style n'est pas défini et `backgroundColor` est utilisé à sa place pour toutes les lignes.

Style	Thème	Description
backgroundColor	Les deux	Couleur d'arrière-plan de la liste. La couleur par défaut est le blanc et est définie sur la déclaration de style de classe. Ce style est ignoré si la valeur de alternatingRowColors est spécifiée.
backgroundDisabledColor	Les deux	Couleur d'arrière-plan lorsque la propriété enabled du composant est définie sur "false". La valeur par défaut est 0xDDDDDD (gris moyen).
borderStyle	Les deux	Le composant List utilise une occurrence RectBorder en tant que bordure et répond aux styles définis pour cette classe. Voir « Classe RectBorder », à la page 1111. La valeur par défaut du style de bordure est "inset".
color	Les deux	Couleur du texte.
disabledColor	Les deux	Couleur du texte lorsque le composant est désactivé. La couleur par défaut est 0x848384 (gris foncé).
embedFonts	Les deux	Valeur booléenne qui indique si la police spécifiée dans fontFamily est de type intégré. Ce style doit être défini sur true si fontFamily fait référence à une police intégrée. Dans le cas contraire, la police intégrée n'est pas utilisée. Si ce style est défini sur true et que fontFamily ne fait pas référence à une police intégrée, aucun texte n'est affiché. La valeur par défaut est false.
fontFamily	Les deux	Nom de la police du texte. La valeur par défaut est "_sans".
fontSize	Les deux	Taille de la police, en points. La valeur par défaut est 10.
fontStyle	Les deux	Style de police : "normal" ou "italic". La valeur par défaut est "normal".
fontWeight	Les deux	Epaisseur de la police : "none" ou "bold". La valeur par défaut est "none". Tous les composants peuvent également accepter la valeur "normal" au lieu de "none" pendant un appel à la méthode setStyle(), mais les appels suivants à getStyle() renvoient "none".
textAlign	Les deux	Alignement du texte : "left", "right" ou "center". La valeur par défaut est "left".

Style	Thème	Description
`textDecoration`	Les deux	Décoration du texte : `"none"` ou `"underline"`. La valeur par défaut est`"none"`.
`textIndent`	Les deux	Nombre indiquant le retrait du texte. La valeur par défaut est 0.
`defaultIcon`	Les deux	Nom de l'icône par défaut à afficher sur chaque ligne. La valeur par défaut est `undefined`, ce qui signifie qu'aucune icône n'est affichée.
`repeatDelay`	Les deux	Nombre de millisecondes d'attente entre le moment où l'utilisateur appuie pour la première fois sur un bouton de la barre défilement et celui où l'action commence à se reproduire. La valeur par défaut est 500, c'est-à-dire une demi-seconde.
`repeatInterval`	Les deux	Nombre de millisecondes séparant les clics automatiques lorsque l'utilisateur maintient le bouton de la souris enfoncé sur un bouton de la barre de défilement. La valeur par défaut est 35.
`rollOverColor`	Les deux	Couleur d'arrière-plan d'une ligne survolée. La couleur par défaut est 0xE3FFD6 (vert vif) pour le thème Halo et 0xAAAAAA (gris clair) pour le thème Sample. Lorsque `themeColor` est modifié par un appel à `setStyle()`, `rollOverColor` est défini sur une valeur liée au style `themeColor` choisi.
`selectionColor`	Les deux	Couleur d'arrière-plan d'une ligne sélectionnée. La couleur par défaut est 0xCDFFC1 (vert clair) pour le thème Halo et 0xEEEEEE (gris très clair) pour le thème Sample. Lorsque `themeColor` est modifié par un appel à `setStyle()`, `selectionColor` est défini sur une valeur liée au style `themeColor` choisi.
`selectionDuration`	Les deux	Durée de la transition entre un état normal à l'état sélectionné et inversement, en millisecondes. La valeur par défaut est 200.
`selectionDisabledColor`	Les deux	Couleur d'arrière-plan d'une ligne sélectionnée. La valeur par défaut est 0xDDDDDD (gris moyen). Etant donné que la valeur par défaut de cette propriété est identique à la valeur par défaut de `backgroundDisabledColor`, la sélection est invisible lorsque le composant est désactivé, à moins que l'une de ces propriétés de style soit modifiée.

Style	Thème	Description
`selectionEasing`	Les deux	Référence à l'équation d'accélération utilisée pour contrôler la transition entre les états de sélection. Ce style ne s'applique qu'à la transition entre l'état normal et l'état sélectionné. L'équation par défaut utilise une formule sine in/out. Pour plus d'informations, reportez-vous à « Personnalisation des animations de composants » dans *Utilisation des composants ActionScript 2.0*.
`textRollOverColor`	Les deux	Couleur du texte lorsque le pointeur passe dessus. La valeur par défaut est 0x2B333C (gris foncé). Ce style est important lors de la définition de `rollOverColor`, car les deux paramètres doivent se compléter l'un l'autre pour que le texte soit bien visible lors d'un survol.
`textSelectedColor`	Les deux	Couleur du texte de la ligne sélectionnée. La valeur par défaut est 0x005F33 (gris foncé). Ce style est important lors de la définition de `selectionColor`, car les deux paramètres doivent se compléter l'un l'autre pour que le texte soit bien visible lors de sa sélection.
`useRollOver`	Les deux	Détermine si le survol d'une ligne active sa mise en surbrillance. La valeur par défaut est `true`.

Définition de styles pour tous les composants List d'un document

La classe List hérite de la classe ScrollSelectList. Les propriétés de style de niveau classe sont définies sur la classe ScrollSelectList, étendue par le composant Menu et tous les composants de type List. Vous pouvez définir de nouvelles valeurs de style par défaut directement sur cette classe, les nouveaux paramètres sont alors reflétés dans tous les composants concernés.

```
_global.styles.ScrollSelectList.setStyle("backgroundColor", 0xFF00AA);
```

Pour ne définir une propriété de style que pour les composants List ou basés sur des listes, vous pouvez créer une occurrence `CSSStyleDeclaration` et la stocker dans `_global.styles.List`.

```
import mx.styles.CSSStyleDeclaration;
if (_global.styles.List == undefined) {
  _global.styles.List = new CSSStyleDeclaration();
}
_global.styles.List.setStyle("backgroundColor", 0xFF00AA);
```

Lors de la création d'une déclaration de style de niveau classe, toutes les valeurs par défaut fournies par la déclaration `ScrollSelectList` sont perdues. Ceci inclut `backgroundColor` qui est nécessaire pour prendre en charge les événements de souris. Pour créer une déclaration de style de niveau de classe et conserver les valeurs par défaut, utilisez une boucle `for..in` pour copier les anciens paramètres dans la nouvelle déclaration.

```
var source = _global.styles.ScrollSelectList;
var target = _global.styles.List;
for (var style in source) {
  target.setStyle(style, source.getStyle(style));
}
```

Pour appliquer des styles au composant List mais pas aux composants qui l'étendent (DataGrid et Tree), vous devez fournir des déclarations de style de niveau classe pour ces sous-classes.

```
import mx.styles.CSSStyleDeclaration;
if (_global.styles.DataGrid == undefined) {
  _global.styles.DataGrid = new CSSStyleDeclaration();
}
_global.styles.DataGrid.setStyle("backgroundColor", 0xFFFFFF);
if (_global.styles.Tree == undefined) {
  _global.styles.Tree = new CSSStyleDeclaration();
}
_global.styles.Tree.setStyle("backgroundColor", 0xFFFFFF);
```

Pour plus d'informations sur les styles de niveau classe, reportez-vous à « Définition de styles pour une classe de composants » dans *Utilisation des composants ActionScript 2.0*.

Utilisation des enveloppes avec le composant List

Le composant List utilise une occurrence de RectBorder pour sa bordure et des barres de défilement pour parcourir les images. Pour plus d'informations sur l'application d'enveloppes à ces éléments visuels, reportez-vous à « Classe RectBorder », à la page 1111 et à « Utilisation d'enveloppes avec le composant UIScrollBar », à la page 1448.

Classe List

Héritage MovieClip > Classe UIObject > Classe UIComponent > View > ScrollView > ScrollSelectList > List

Nom de classe ActionScript mx.controls.List

Le composant List comprend trois parties : des éléments, des lignes et un fournisseur de données.

Un *élément* est un objet ActionScript utilisé pour stocker les unités d'informations dans la liste. Une liste peut être considérée comme un tableau ; chaque espace indexé du tableau constitue un élément. Un élément est un objet qui dispose, en règle générale, d'une propriété `label` affichée et d'une propriété `data` utilisée pour stocker des données.

Une *ligne* est un composant utilisé pour afficher un élément. Les lignes sont fournies par défaut par la liste (avec la classe SelectableRow), mais vous pouvez également les fournir sous forme de sous-classes de la classe SelectableRow. La classe SelectableRow implémente l'API CellRenderer, ensemble des propriétés et méthodes qui permettent à la liste de manipuler toutes les lignes et d'envoyer des données et des informations d'état (par exemple, taille, sélections, etc.) à la ligne pour l'affichage.

Le fournisseur de données correspond au modèle de données des éléments d'une liste. Un tableau situé dans la même image qu'une liste reçoit automatiquement des méthodes qui permettent de manipuler les données et de diffuser les changements vers plusieurs affichages. Vous pouvez créer une occurrence de tableau ou en obtenir une auprès d'un serveur et l'utiliser comme modèle de données pour plusieurs composants List, ComboBox, DataGrid, etc. Le composant List dispose de méthodes agissant comme proxy pour son fournisseur de données (par exemple `addItem()` et `removeItem()`). Si vous ne fournissez aucun fournisseur de données externe à la liste, ces méthodes créent automatiquement une occurrence DataProvider, exposée par le biais de `List.dataProvider`.

Pour ajouter un composant List à l'ordre des tabulations d'une application, définissez sa propriété `tabIndex` (voir `UIComponent.tabIndex`). Le composant List utilise le gestionnaire de focus pour remplacer le rectangle de focus par défaut de Flash Player et tracer un rectangle de focus personnalisé aux coins arrondis. Pour plus d'informations, consultez « Création de la navigation personnalisée du focus » dans *Utilisation des composants ActionScript 2.0*.

Toutes les classes de composants ont une propriété `version` qui correspond à une propriété de classe. Les propriétés de classe ne sont disponibles que dans la classe elle-même. La propriété `version` renvoie une chaîne qui indique la version du composant. Pour accéder à cette propriété, utilisez le code suivant :

```
trace(mx.controls.List.version);
```

REMARQUE

Le code `trace(myListInstance.version);` renvoie `undefined`.

Méthodes de la classe List

Le tableau suivant présente les méthodes de la classe List.

Méthode	Description
`List.addItem()`	Ajoute un élément à la fin de la liste.
`List.addItemAt()`	Ajoute un élément à la liste, à l'index spécifié.
`List.getItemAt()`	Renvoie l'élément à l'emplacement d'index spécifié.
`List.removeAll()`	Supprime tous les éléments de la liste.
`List.removeItemAt()`	Supprime l'élément à l'index spécifié.
`List.replaceItemAt()`	Remplace l'élément par un autre, à l'index spécifié.
`List.setPropertiesAt()`	Applique les propriétés spécifiées à l'élément donné.
`List.sortItems()`	Trie les éléments de la liste à l'aide de la fonction de comparaison spécifiée.
`List.sortItemsBy()`	Trie les éléments de la liste à l'aide d'une propriété donnée.

Méthodes héritées de la classe UIObject

Le tableau suivant énumère les méthodes de la classe List héritées de la classe UIObject. Pour appeler ces méthodes, utilisez le formulaire *`listInstance.methodName`*.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
`UIObject.setSize()`	Redimensionne l'objet à la taille demandée.
`UIObject.setSkin()`	Définit une enveloppe dans l'objet.
`UIObject.setStyle()`	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant énumère les méthodes de la classe List héritées de la classe UIComponent. Pour appeler ces méthodes, utilisez le formulaire *listInstance.methodName*.

Méthode	Description
`UIComponent.getFocus()`	Renvoie une référence à l'objet ayant le focus.
`UIComponent.setFocus()`	Attribue le focus à l'occurrence de composant.

Propriétés de la classe List

Le tableau suivant présente les propriétés de la classe List.

Propriété	Description
`List.cellRenderer`	Affecte la classe ou le symbole à utiliser pour afficher chaque ligne de la liste.
`List.dataProvider`	Source des éléments de la liste.
`List.hPosition`	Position horizontale de la liste.
`List.hScrollPolicy`	Indique si la barre de défilement horizontale est affichée (`"on"`) ou non (`"off"`).
`List.iconField`	Champ situé dans chaque élément pour désigner les icônes.
`List.iconFunction`	Fonction qui détermine les icônes à utiliser.
`List.labelField`	Spécifie un champ dans chaque élément, à utiliser comme texte d'étiquette.
`List.labelFunction`	Fonction qui détermine les champs de chaque élément à utiliser pour le texte d'étiquette.
`List.length`	Nombre d'éléments de la liste. Cette propriété est en lecture seule.
`List.maxHPosition`	Nombre de pixels que la liste peut faire défiler à droite, lorsque `List.hScrollPolicy` est défini sur `"on"`.
`List.multipleSelection`	Indique si la sélection multiple est autorisée dans la liste (`true`) ou non (`false`).
`List.rowCount`	Nombre de lignes au moins partiellement visibles dans la liste.
`List.rowHeight`	Hauteur de chaque ligne de la liste, en pixels.
`List.selectable`	Indique si la liste peut être sélectionnée (`true`) ou non (`false`).
`List.selectedIndex`	Index d'une sélection dans une liste à sélection unique.

Propriété	Description
`List.selectedIndices`	Tableau des éléments sélectionnés dans une liste à sélection multiple.
`List.selectedItem`	Elément sélectionné dans une liste à sélection unique. Cette propriété est en lecture seule.
`List.selectedItems`	Objets sélectionnés dans une liste à sélection multiple. Cette propriété est en lecture seule.
`List.vPosition`	Premier élément visible de la liste.
`List.vScrollPolicy`	Indique si la barre de défilement verticale est affichée (`"on"`), ne l'est pas (`"off"`) ou est affichée si nécessaire (`"auto"`).

Propriétés héritées de la classe UIObject

Le tableau suivant énumère les propriétés de la classe List héritées de la classe UIObject. Pour accéder à ces propriétés, utilisez le formulaire *`listInstance.propertyName`*.

Propriété	Description
`UIObject.bottom`	Position du bord inférieur de l'objet par rapport au bord inférieur de son parent. Lecture seule.
`UIObject.height`	Hauteur de l'objet, en pixels. Lecture seule.
`UIObject.left`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.right`	Position du bord droit de l'objet par rapport au bord droit de son parent. Lecture seule.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Position du bord supérieur de l'objet par rapport à son parent. Lecture seule.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Largeur de l'objet, en pixels. Lecture seule.
`UIObject.x`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.y`	Bord supérieur de l'objet, en pixels. Lecture seule.

Propriétés héritées de la classe UIComponent

Le tableau suivant énumère les propriétés de la classe List héritées de la classe UIComponent. Pour accéder à ces propriétés, utilisez le formulaire *listInstance.propertyName*.

Propriété	Description
UIComponent.enabled	Indique si le composant peut recevoir le focus et la saisie.
UIComponent.tabIndex	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Evénements de la classe List

Le tableau suivant présente les événements de la classe List.

Evénement	Description
List.change	Diffusé chaque fois que la sélection change suite à une interaction avec l'utilisateur.
List.itemRollOut	Diffusé lorsque le pointeur survole les éléments de la liste, puis cesse de les survoler.
List.itemRollOver	Diffusé lorsque le pointeur passe au-dessus des éléments de la liste.
List.scroll	Diffusé lorsqu'une liste défile.

Evénements hérités de la classe UIObject

Le tableau suivant énumère les événements de la classe List hérités de la classe UIObject.

Evénement	Description
UIObject.draw	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
UIObject.hide	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
UIObject.load	Diffusé lorsque des sous-objets sont créés.
UIObject.move	Diffusé lorsque l'objet a été déplacé.
UIObject.resize	Diffusé lorsqu'un objet a été redimensionné.
UIObject.reveal	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
UIObject.unload	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant énumère les événements de la classe List hérités de la classe UIComponent.

Evénement	Description
`UIComponent.focusIn`	Diffusé lorsqu'un objet reçoit le focus.
`UIComponent.focusOut`	Diffusé lorsqu'un objet perd le focus.
`UIComponent.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIComponent.keyUp`	Diffusé lorsqu'une touche est relâchée.

List.addItem()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.addItem(*label*[, *data*])

listInstance.addItem(*itemObject*)

Paramètres

label Chaîne indiquant l'étiquette du nouvel élément.

data Données de l'élément. Ce paramètre est facultatif et peut être de n'importe quel type de données.

itemObject Objet d'élément possédant généralement les propriétés `label` et `data`.

Valeur renvoyée

Index auquel l'élément a été ajouté.

Description

Méthode qui ajoute un nouvel élément à la fin de la liste.

Dans le premier exemple d'utilisation, un objet d'élément est toujours créé avec la propriété `label` spécifiée et, le cas échéant, la propriété `data`.

Le second exemple d'utilisation ajoute l'objet spécifié.

L'appel à cette méthode modifie le fournisseur de données du composant List. Si le fournisseur de données est partagé par d'autres composants, ceux-ci sont également mis à jour.

Exemple

Les deux lignes de code suivantes ajoutent un élément à l'occurrence `my_list`. Pour tester ce code, faites glisser un composant List sur la scène et nommez l'occurrence **my_list**. Ajoutez le code suivant à l'Image 1 du scénario :

```
var my_list:mx.controls.List;

my_list.addItem("this is an Item");
my_list.addItem({label:"Gordon", age:"very old", data:123});
```

List.addItemAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.addItemAt(*index*, *label*[, *data*])

listInstance.addItemAt(*index*, *itemObject*)

Paramètres

index Nombre supérieur ou égal à 0 qui indique la position de l'élément.

label Chaîne indiquant l'étiquette du nouvel élément.

data Données de l'élément. Ce paramètre est facultatif et peut être de n'importe quel type de données.

itemObject Objet d'élément possédant généralement les propriétés `label` et `data`.

Valeur renvoyée

Index auquel l'élément a été ajouté.

Description

Méthode : ajoute un nouvel élément à la position spécifiée par le paramètre *index*.

Dans le premier exemple d'utilisation, un objet d'élément est toujours créé avec la propriété `label` spécifiée et, le cas échéant, la propriété `data`.

Le second exemple d'utilisation ajoute l'objet spécifié.

L'appel à cette méthode modifie le fournisseur de données du composant List. Si le fournisseur de données est partagé par d'autres composants, ceux-ci sont également mis à jour.

Exemple

L'exemple suivant ajoute un élément à la première position d'index, qui correspond au deuxième élément dans la liste. Pour tester ce code, faites glisser un composant List sur la scène et nommez l'occurrence **my_list**. Ajoutez le code suivant à l'Image 1 du scénario :

```
var my_list:mx.controls.List;

my_list.addItem("this is an Item");
my_list.addItem({label:"Gordon", age:"very old", data:123});
my_list.addItemAt(1, {label:"Red", data:0xFF0000});
```

List.cellRenderer

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.cellRenderer

Description

Propriété qui affecte l'objet cellRenderer à utiliser pour chaque ligne de la liste.
Cette propriété doit correspondre à une référence d'objet de classe ou à un identifiant de liaison de symbole. Toutes les classes utilisées pour cette propriété doivent implémenter API CellRenderer.

Exemple

L'exemple suivant utilise un identifiant de liaison pour définir un nouveau rendu de cellule :

```
my_list.cellRenderer = "ComboBoxCell";
```

List.change

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.change = function(eventObject:Object) {
  // Votre code ici.
};
listInstance.addEventListener("change", listenerObject);
```

Utilisation 2 :

```
on (change) {
  // Votre code ici.
}
```

Description

Evénement, diffusé à tous les écouteurs enregistrés lorsque l'index sélectionné dans la liste change suite à l'interaction d'un utilisateur.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence de composant (*listInstance*) distribue un événement (ici, `change`) qui est géré par une fonction, également appelée un *gestionnaire*, associée à un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour plus d'informations, voir « Classe EventDispatcher (API) », à la page 517.

Pour finir, vous appelez la méthode `addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour associer l'écouteur à l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Le second exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence du composant List. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence du composant. Par exemple, le code suivant, lié à l'occurrence du composant List `my_list`, envoie « _level0.my_list » dans le panneau Sortie :

```
on (change) {
  trace(this);
}
```

Exemple

L'exemple suivant ajoute trois éléments au composant List. Modifiez la valeur sélectionnée de la liste pour que la valeur de l'élément nouvellement sélectionné s'affiche dans le panneau Sortie. Pour tester ce code, faites glisser un composant List sur la scène et nommez l'occurrence **my_list**. Ajoutez le code suivant à l'image 1 du scénario :

```
var my_list:mx.controls.List;

my_list.addItem({data:'flash', label:'Flash'});
my_list.addItem({data:'dreamweaver', label:'Dreamweaver'});
my_list.addItem({data:'coldfusion', label:'ColdFusion'});

// Création d'un objet écouteur.
var listListener:Object = new Object();
listListener.change = function(evt_obj:Object) {
   trace("Value changed to: " + evt_obj.target.value);
}
// Ajout de l'écouteur.
my_list.addEventListener("change", listListener);
```

Voir aussi

`EventDispatcher.addEventListener()`

List.dataProvider

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`listInstance``.dataProvider`

Description

Propriété, modèle de données pour les éléments affichés dans une liste. La valeur de cette propriété peut être un tableau ou tout objet qui implémente l'API DataProvider. La valeur par défaut est `[]`. Pour plus d'informations, voir « API DataProvider », à la page 333.

Le composant List, comme d'autres composants de données, ajoute des méthodes au prototype de l'objet tableau pour être conforme à l'API DataProvider. De ce fait, tout tableau existant parallèlement en tant que liste dispose automatiquement de toutes les méthodes (`addItem()`, `getItemAt()`, etc.) nécessaires pour être le modèle de données d'une liste et peut être utilisé pour diffuser les changements de modèle vers plusieurs composants.

Si le tableau contient des objets, l'utilisateur accède aux propriétés `List.labelField` ou `List.labelFunction` pour déterminer les parties de l'élément à afficher. La valeur par défaut est « `label` », ce qui signifie que lorsqu'un champ `label` existe, il est affiché. S'il n'existe pas, la liste de tous les champs séparés par une virgule est affichée.

REMARQUE

Si le tableau contient des chaînes dans tous les emplacements d'index et aucun objet, la liste ne peut pas trier les éléments et conserver l'état de sélection. Tout tri entraîne la perte de la sélection.

Toute occurrence implémentant l'API DataProvider peut agir comme fournisseur de données pour un composant List. Cela inclut Flash Remoting RecordSets, Firefly DataSets, etc.

Exemple

L'exemple suivant utilise un tableau de chaînes destiné à remplir la liste :

```
my_list.dataProvider = ["Ground Shipping", "2nd Day Air", "Next Day Air"];
```

Cet exemple crée un tableau fournisseur de données et lui affecte la propriété `dataProvider`, comme suit :

```
var myDP_array:Array = new Array();
my_list.dataProvider = myDP_array;

var accounts_array:Array = new Array();
accounts_array.push({name:"checkings", accountID:12345});
accounts_array.push({name:"savings", accountID:67890});

for (var i:Number = 0; i < accounts_array.length; i++) {
 // Ces modifications apportées au fournisseur de données seront
 // diffusées dans la liste.
 myDP_array.addItem({label:accounts_array[i].name,
  data:accounts_array[i].accountID});
}
```

List.getItemAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
listInstance.getItemAt(index)
```

Paramètres

index Nombre supérieur ou égal à 0 et inférieur à `List.length`. Ce nombre désigne l'index de l'élément à récupérer.

Valeur renvoyée

Objet de l'élément indexé ; `undefined` si l'index est hors limites.

Description

Méthode qui récupère l'élément à l'emplacement d'index spécifié. Cette méthode permet d'obtenir l'élément de données à partir d'un tableau, du composant DataProvider ou d'un objet de données créé avec `CellRenderer.setValue()`.

Exemple

Le code suivant affiche l'étiquette de l'élément à la position d'index 2. Pour essayer ce code, faites glisser un composant List sur la scène et nommez l'occurrence **my_list**. Ajoutez le code suivant à l'Image 1 du scénario :

```
var my_list:mx.controls.List;

my_list.addItem({data:'flash', label:'Flash'});
my_list.addItem({data:'dreamweaver', label:'Dreanweaver'});
my_list.addItem({data:'coldfusion', label:'ColdFusion'});

trace(my_list.getItemAt(2).label);
```

List.hPosition

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.hPosition

Description

Propriété qui fait défiler la liste horizontalement en fonction du nombre de pixels spécifié. Vous ne pouvez pas définir `hPosition` à moins que la valeur de `hScrollPolicy` soit définie sur « `on` » et que la valeur de `maxHPosition` soit supérieure à 0.

Exemple

Le code suivant affiche la valeur actuelle de `hPosition` lorsque vous faites défiler horizontalement l'occurrence du composant List. Pour tester ce code, faites glisser un composant List sur la scène et nommez l'occurrence **my_list**. Ajoutez le code suivant à l'Image 1 du scénario :

```
var my_list:mx.controls.List;

my_list.setSize(150, 100);
my_list.hScrollPolicy = "on";
my_list.maxHPosition = 50;

my_list.addItem({data:'flash', label:'Flash'});
my_list.addItem({data:'dreamweaver', label:'Dreanweaver'});
my_list.addItem({data:'coldfusion', label:'ColdFusion'});

var listListener:Object = new Object();
listListener.scroll = function (evt_obj:Object) {
 trace("my_list.hPosition = " + my_list.hPosition);
}
my_list.addEventListener("scroll", listListener);
```

List.hScrollPolicy

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.hScrollPolicy

Description

Propriété : chaîne qui détermine si la barre de défilement horizontale est affichée (« on ») ou non (« off »). La valeur par défaut est « off ». La barre de défilement horizontale ne mesure pas le texte. Vous devez définir une position de défilement horizontale maximale (voir List.maxHPosition).

REMARQUE	List.hScrollPolicy ne reconnaît pas la valeur "auto".

Exemple

Le code suivant permet à la liste de faire défiler horizontalement jusqu'à 200 pixels. Pour tester ce code, faites glisser un composant List sur la scène et nommez l'occurrence **my_list**. Ajoutez le code suivant à l'image 1 du scénario :

```
var my_list:mx.controls.List;

my_list.setSize(150, 100);
my_list.hScrollPolicy = "on";
my_list.maxHPosition = 200;

my_list.addItem({data:'flash', label:'Flash'});
my_list.addItem({data:'dreamweaver', label:'Dreanweaver'});
my_list.addItem({data:'coldfusion', label:'ColdFusion'});
```

Voir aussi

List.hPosition, List.maxHPosition

List.iconField

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.iconField

Description

Propriété qui spécifie le nom d'un champ à utiliser en tant qu'identifiant d'icône. Si la valeur du champ est `undefined`, l'icône par défaut spécifiée par le style `defaultIcon` est utilisée. Si le style `defaultIcon` est `undefined`, aucune icône n'est utilisée.

Exemple

L'exemple suivant définit la propriété `iconField` sur la propriété `icon` de chaque élément. Pour tester ce code, faites glisser un composant List sur la scène, nommez l'occurrence **my_list** et créez respectivement trois symboles avec les noms d'occurrence `flash`, `dreamweaver` et `cf`. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant List sur la scène (nom d'occurrence : my_list)
  - Symbole MovieClip/Graphic dans la bibliothèque avec l'identifiant de
  liaison « flash »
  - Symbole MovieClip/Graphic dans la bibliothèque avec l'identifiant de
  liaison « dreamweaver »
  - Symbole MovieClip/Graphic dans la bibliothèque avec l'identifiant de
  liaison « cf »
*/

var my_list:mx.controls.List;

my_list.setSize(200, 100);

my_list.addItem({data:"flash", label:"Flash", icon:"flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver",
  icon:"dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion", icon:"cf"});

my_list.iconField = "icon";
```

Voir aussi

`List.iconFunction`

List.iconFunction

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.iconFunction

Description

Propriété qui spécifie une fonction déterminant l'icône à utiliser pour afficher l'élément de chacune des lignes. Cette fonction reçoit un paramètre, *item*, qui correspond à l'élément restitué et doit renvoyer une chaîne représentant l'identificateur de symbole de l'icône.

Exemple

L'exemple suivant ajoute des icônes qui indiquent si un fichier correspond à un document image ou texte. Si le champ `data.fileExtension` contient une valeur `"jpg"` ou `"gif"`, l'icône utilisée est `"pictureIcon"`, etc. :

```
my_list.iconFunction = function(item:Object):String {
  var type:String = item.data.fileExtension;
  if (type == "jpg" || type == "gif") {
    return "pictureIcon";
  } else if (type == "doc" || type == "txt") {
    return "docIcon";
  }
}
```

L'exemple suivant définit la propriété `iconField` sur la propriété `icon` de chaque élément. Pour tester ce code, faites glisser un composant List sur la scène, nommez l'occurrence **my_list** et créez respectivement trois symboles avec les noms d'occurrence `flash`, `dreamweaver` et `cf`. Ajoutez le code suivant à l'Image 1 du scénario :

```
/**
 Requiert :
  - Composant List sur la scène (nom d'occurrence : my_list)
  - Symbole MovieClip/Graphic dans la bibliothèque avec l'identifiant de
  liaison « flashIcon »
*/

var my_list:mx.controls.List;

my_list.setSize(200, 100);
```

```
my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

my_list.iconFunction = function(item:Object):String {
  if (item.data == "flash") {
    // Placement d'une icône en regard de l'élément de liste avec
    // les données « flash »
    return "flashIcon";
  }
};
my_list.iconField = "icon";
```

List.itemRollOut

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.itemRollOut = function(eventObject:Object) {
  // Votre code ici.
};
listInstance.addEventListener("itemRollOut", listenerObject);
```

Utilisation 2 :

```
on (itemRollOut) {
  // Votre code ici.
}
```

Objet événement

Outre les propriétés standard de l'objet événement, l'événement `itemRollOut` possède une propriété `index` qui indique le numéro de l'élément survolé.

Description

Evénement ; diffusé à tous les écouteurs enregistrés lorsque le pointeur survole, puis cesse de survoler, les éléments de la liste.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence de composant (*listInstance*) distribue un événement (ici, itemRollOut) qui est géré par une fonction, également appelée un *gestionnaire*, associée à un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode EventDispatcher.addEventListener() sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Pour plus d'informations, voir « Classe EventDispatcher (API) », à la page 517.

Le second exemple d'utilisation fait appel au gestionnaire on() et doit être associé directement à une occurrence du composant List. Le mot-clé this, utilisé dans un gestionnaire on() lié à un composant, correspond à l'occurrence du composant. Par exemple, le code suivant, lié à l'occurrence du composant List my_list, envoie « _level0.my_list » dans le panneau Sortie :

```
on (itemRollOut) {
  trace(this);
}
```

Exemple

L'exemple suivant envoie un message vers le panneau Sortie qui indique le numéro d'index survolé par le pointeur :

```
var my_list:mx.controls.List;

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

// Création d'un objet écouteur.
var listListener:Object = new Object();
listListener.itemRollOut = function(evt_obj:Object) {
 trace("Item #" + evt_obj.index + " has been rolled out.");
};

// Ajout de l'écouteur.
my_list.addEventListener("itemRollOut", listListener);
```

Voir aussi

List.itemRollOver

List.itemRollOver

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.itemRollOver = function(eventObject:Object) {
  // Votre code ici.
};
listInstance.addEventListener("itemRollOver", listenerObject);
```

Utilisation 2 :

```
on (itemRollOver) {
  // Votre code ici.
}
```

Objet événement

Outre les propriétés standard de l'objet événement, l'événement `itemRollOver` possède une propriété `index` qui précise le numéro de l'élément survolé.

Description

Evénement, diffusé à tous les écouteurs enregistrés lorsque les éléments de la liste sont survolés.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence de composant (*`listInstance`*) distribue un événement (ici, `itemRollOver`) qui est géré par une fonction, également appelée un *gestionnaire*, associée à un objet écouteur (*`listenerObject`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`eventObject`*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Pour plus d'informations, voir « Classe EventDispatcher (API) », à la page 517.

Le second exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence du composant List. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence du composant. Par exemple, le code suivant, lié à l'occurrence du composant List `my_list`, envoie « _level0.my_list » dans le panneau Sortie :

```
on (itemRollOver) {
  trace(this);
}
```

Exemple

L'exemple suivant envoie un message vers le panneau Sortie qui indique le numéro d'index survolé par le pointeur :

```
var my_list:mx.controls.List;

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

// Création d'un objet écouteur.
var listListener:Object = new Object();
listListener.itemRollOver = function (evt_obj:Object) {
 trace("Item #" + evt_obj.index + " has been rolled over.");
};

// Ajout de l'écouteur.
my_list.addEventListener("itemRollOver", listListener);
```

Voir aussi

`List.itemRollOut`

List.labelField

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.labelField

Description

Propriété qui spécifie, dans chaque élément, un champ à utiliser comme texte d'affichage. Cette propriété prend la valeur du champ et l'utilise comme étiquette. La valeur par défaut est « label ».

Exemple

L'exemple suivant définit la propriété labelField sur le champ « name » de chaque élément. L'élément ajouté à la deuxième ligne de code aurait « Nina » comme étiquette :

```
var my_list:mx.controls.List;

my_list.labelField = "name";
my_list.addItem({name: "Nina", age: 25});
```

Voir aussi

List.labelFunction

List.labelFunction

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.labelFunction

Description

Propriété qui spécifie une fonction déterminant le champ (ou la combinaison de champs) à afficher pour chaque élément. Cette fonction reçoit un paramètre, *item*, qui correspond à l'élément restitué et doit renvoyer une chaîne représentant le texte à afficher.

Exemple

L'exemple suivant affiche des détails formatés des éléments dans l'étiquette :

```
var my_list:mx.controls.List;

my_list.setSize(300, 100);

// Définition de la façon dont les données de la liste s'afficheront.
my_list.labelFunction = function(item_obj:Object):String {
 var label_str:String = item_obj.label + " - Code is: " + item_obj.data;
 return label_str;
}

// Ajout des données à la liste.
my_list.addItem({data:"f", label:"Flash"});
my_list.addItem({data:"d", label:"Dreamweaver"});
my_list.addItem({data:"c", label:"ColdFusion"});
```

Voir aussi

`List.labelField`

List.length

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.length

Description

Propriété (lecture seule), nombre d'éléments dans la liste.

Exemple

L'exemple suivant affiche le nombre d'éléments se trouvant dans le fournisseur de données de la liste :

```
var my_list:mx.controls.List;

// Ajout des données à la liste.
my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

var listLength_num:Number = my_list.length;
trace("Length of List: " + listLength_num);
```

List.maxHPosition

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.maxHPosition

Description

Propriété ; spécifie le nombre de pixels que la liste peut faire défiler lorsque `List.hScrollPolicy` est défini sur `"on"`. La liste ne mesure pas précisément la largeur du texte qu'elle contient. Vous devez définir `maxHPosition` pour indiquer le volume de défilement requis. Si cette propriété n'est pas définie, la liste ne défile pas horizontalement.

Exemple

L'exemple suivant crée une liste avec 200 pixels de défilement horizontal :

```
var my_list:mx.controls.List;

my_list.setSize(150, 100);
my_list.hScrollPolicy = "on";
my_list.maxHPosition = 200;

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});
```

Voir aussi

`List.hScrollPolicy`

List.multipleSelection

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.multipleSelection

Description

Propriété : indique si les sélections autorisées sont multiples (`true`) ou uniques (`false`). La valeur par défaut est `false`.

Exemple

L'exemple suivant teste pour déterminer si plusieurs éléments peuvent être sélectionnés et si c'est le cas, affiche des instructions dans un composant Label. Pour tester ce code, faites glisser un composant List sur la scène et nommez l'occurrence **my_list**. Faites ensuite glisser un composant Label sur la scène et nommez l'occurrence **my_label**. Ajoutez le code suivant à l'Image 1 du scénario :

```
var my_list:mx.controls.List;

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

my_list.multipleSelection = true;

if (my_list.multipleSelection) {
  my_label.text = "Hold down Control or Shift to select multiple items";
  my_label.autoSize = "left";
}
```

List.removeAll()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
listInstance.removeAll()
```

Valeur renvoyée

Aucune.

Description

Méthode qui supprime tous les éléments de la liste.

L'appel à cette méthode modifie le fournisseur de données du composant List. Si le fournisseur de données est partagé par d'autres composants, ceux-ci sont également mis à jour.

Exemple

Le code suivant efface tous les éléments dans un composant List lorsque vous cliquez sur un bouton. Pour tester ce code, faites glisser un composant List sur la scène et nommez l'occurrence **my_list**. Faites ensuite glisser un composant Button sur la scène et nommez l'occurrence **remove_button**. Ajoutez le code suivant à l'Image 1 du scénario :

```
var my_list:mx.controls.List;
var remove_button:mx.controls.Button;

remove_button.label = "Remove";

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object) {
  my_list.removeAll();
  evt_obj.target.enabled = false;
}
remove_button.addEventListener("click", buttonListener);
```

List.removeItemAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.removeItemAt(*index*)

Paramètres

index Numéro indiquant la position de l'élément. La valeur doit être supérieure à 0 et inférieure à List.length.

Valeur renvoyée

Un objet : l'élément supprimé (undefined si aucun élément n'existe).

Description

Méthode qui supprime l'élément situé à l'emplacement d'index indiqué. L'index suivant l'index spécifié disparaît.

L'appel à cette méthode modifie le fournisseur de données du composant List. Si le fournisseur de données est partagé par d'autres composants, ceux-ci sont également mis à jour.

Exemple

Le code suivant efface l'élément sélectionné dans un composant List lorsque vous cliquez sur un bouton. Pour tester ce code, faites glisser un composant List sur la scène et nommez l'occurrence **my_list**. Faites ensuite glisser un composant Button sur la scène et nommez l'occurrence **remove_button**. Ajoutez le code suivant à l'Image 1 du scénario :

```
var my_list:mx.controls.List;
var remove_button:mx.controls.Button;

remove_button.label = "Remove";

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object) {
  if (my_list.selectedIndex != undefined) {
    my_list.removeItemAt(my_list.selectedIndex);
  }
}
remove_button.addEventListener("click", buttonListener);
```

List.replaceItemAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
listInstance.replaceItemAt(index, label[, data])
listInstance.replaceItemAt(index, itemObject)
```

Paramètres

index Nombre supérieur à 0 et inférieur à `List.length` qui indique la position à laquelle insérer l'élément (index du nouvel élément).

label Chaîne indiquant l'étiquette du nouvel élément.

data Données de l'élément. Ce paramètre est facultatif et peut être de n'importe quel type.

itemObject Objet à utiliser en tant qu'élément. Il possède généralement les propriétés `label` et `data`.

Valeur renvoyée

Aucune.

Description

Méthode qui remplace le contenu de l'élément à l'emplacement d'index spécifié.

L'appel à cette méthode modifie le fournisseur de données du composant List. Si le fournisseur de données est partagé par d'autres composants, ceux-ci sont également mis à jour.

Exemple

L'exemple suivant remplace l'élément à la position actuellement sélectionnée. Pour tester ce code, faites glisser un composant List sur la scène et nommez l'occurrence **my_list**. Faites ensuite glisser un composant Button sur la scène et nommez l'occurrence **replace_button**. Ajoutez le code suivant à l'Image 1 du scénario :

```
var my_list:mx.controls.List;
var replace_button:mx.controls.Button;

replace_button.label = "Replace";
```

```
my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object) {
  if (my_list.selectedIndex != undefined) {
    my_list.replaceItemAt(my_list.selectedIndex, {data:"flex",
  label:"Flex"});
  }
}
replace_button.addEventListener("click", buttonListener);
```

List.rowCount

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.rowCount

Description

Propriété, nombre de lignes au moins partiellement visibles dans la liste. Ceci s'avère utile si vous avez dimensionné une liste en pixels et devez compter ses lignes. Inversement, la définition du nombre de lignes garantit qu'un nombre exact de lignes est affiché, sans ligne partielle en bas.

Le code `my_list.rowCount = num` équivaut au code `my_list.setSize(my_list.width, h)` (où `h` correspond à la hauteur requise pour afficher les éléments `num`).

La valeur par défaut est basée sur la hauteur de la liste, telle qu'elle est définie au cours de la création ou par la méthode `list.setSize()` (voir `UIObject.setSize()`).

Exemple

L'exemple suivant montre le nombre d'éléments visibles dans une liste :

```
var rowCount = my_list.rowCount;
```

L'exemple suivant permet d'afficher quatre éléments dans la liste :

```
my_list.rowCount = 4;
```

L'exemple suivant supprime une ligne partielle en bas d'une liste, le cas échéant :

```
my_list.rowCount = my_list.rowCount;
```

L'exemple suivant définit une liste au plus petit nombre de lignes qu'elle peut afficher entièrement :

```
my_list.rowCount = 1;
trace("my_list has " + my_list.rowCount + " rows");
```

L'exemple suivant redimensionne la liste à l'aide de la méthode `setSize()`, puis définit un compte de lignes de 8 éléments :

```
my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

my_list.setSize(200, 30);
my_list.rowCount = 8;
trace("my_list has " + my_list.rowCount + " rows.");
```

List.rowHeight

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`listInstance``.rowHeight`

Description

Propriété, hauteur de chaque ligne dans la liste, en pixels. Les paramètres de la police n'agrandissent pas les lignes pour qu'elles s'adaptent en conséquence. La définition de la propriété `rowHeight` est donc la meilleure façon de s'assurer que les éléments sont affichés dans leur intégralité. La valeur par défaut est 20.

Exemple

L'exemple suivant définit les lignes à 30 pixels :

```
my_list.rowHeight = 30;
```

L'exemple suivant définit la hauteur de chaque ligne sur 30 pixels puis redimensionne la liste en fonction du nombre total d'éléments qu'elle contient :

```
my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

my_list.rowHeight = 30;
my_list.rowCount = my_list.length;
```

List.scroll

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.scroll = function(eventObject:Object) {
  // Votre code ici.
};
listInstance.addEventListener("scroll", listenerObject);
```

Utilisation 2 :

```
on (scroll) {
  // Votre code ici.
}
```

Objet événement

Outre les propriétés standard de l'objet événement, l'événement `scroll` possède une propriété supplémentaire : `direction`. Il s'agit d'une chaîne ayant deux valeurs possibles : « `horizontal` » ou « `vertical` ». Pour un événement scroll ComboBox, la valeur est toujours « `vertical` ».

Description

Evénement : diffusé à tous les écouteurs enregistrés lors du défilement d'une liste.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence de composant (*listInstance*) distribue un événement (ici, `scroll`) géré par une fonction, également appelée *gestionnaire*, associée à un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Pour plus d'informations, voir « Classe EventDispatcher (API) », à la page 517.

Le second exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence du composant List. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence du composant. Par exemple, le code suivant, lié à l'occurrence du composant List `my_list`, envoie « _level0.my_list » dans le panneau Sortie :

```
on (scroll) {
  trace(this);
}
```

Exemple

L'exemple suivant envoie la direction et la position de la liste chaque fois que vous faites défiler ses éléments :

```
var my_list:mx.controls.List;

my_list.rowCount = 2;
for (var i:Number = 0; i < 10; i++) {
  my_list.addItem({data:i, label:"Item #" + i});
}

var listListener:Object = new Object();
listListener.scroll = function(evt_obj:Object) {
  trace("list scrolled (direction:" + evt_obj.direction + ", position:" +
  evt_obj.position + ")");
};
my_list.addEventListener("scroll", listListener);
```

List.selectable

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.selectable

Description

Propriété : valeur booléenne indiquant si la liste est sélectionnable (`true`) ou non (`false`). La valeur par défaut est `true`.

Exemple

L'exemple suivant empêche les utilisateurs de sélectionner des éléments de la liste en définissant la propriété selectable sur `false` :

```
var my_list:mx.controls.List;

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

my_list.selectable = false;
```

List.selectedIndex

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.selectedIndex

Description

Propriété, index sélectionné d'une liste à sélection unique. La valeur est `undefined` si rien n'est sélectionné. Elle est égale au dernier élément sélectionné s'il y a plusieurs sélections. Si vous affectez une valeur à `selectedIndex`, toute sélection en cours est effacée et l'élément indiqué est sélectionné.

L'utilisation de la propriété `selectedIndex` pour modifier la sélection ne diffuse pas d'événement `change`. Pour le distribuer, utilisez le code suivant :

```
my_list.dispatchEvent({type:"change", target:my_list});
```

Exemple

L'exemple suivant sélectionne le premier élément d'une liste par défaut et affiche l'index de l'élément sélectionné chaque fois que l'utilisateur sélectionne un nouvel élément :

```
var my_list:mx.controls.List;

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

// Sélection du premier élément par défaut.
my_list.selectedIndex = 0;

var listListener:Object = new Object();
listListener.change = function(evt_obj:Object) {
  trace("selectedIndex = " + evt_obj.target.selectedIndex);
}
my_list.addEventListener("change", listListener);
```

Voir aussi

`List.selectedIndices`, `List.selectedItem`, `List.selectedItems`

List.selectedIndices

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.selectedIndices

Description

Propriété, tableau des index des éléments sélectionnés. L'affectation de cette propriété remplace la sélection actuelle. La définition de `selectedIndices` sur un tableau de longueur 0 (ou `undefined`) efface la sélection en cours. La valeur est `undefined` si rien n'est sélectionné.

La propriété `selectedIndices` reflète l'ordre de sélection des éléments. Si vous cliquez successivement sur les deuxième, troisième et premier éléments, `selectedIndices` renvoie `[1,2,0]`.

Exemple

L'exemple suivant récupère les index sélectionnés :

```
var selIndices:Array = my_list.selectedIndices;
```

L'exemple suivant sélectionne quatre éléments :

```
var my_array = new Array (1, 4, 5, 7);
my_list.selectedIndices = my_array;
```

L'exemple suivant sélectionne deux éléments de liste par défaut et affiche leur propriété label dans le panneau de sortie :

```
my_list.multipleSelection = true;

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

my_list.selectedIndices = [0, 2];

var numSelected:Number = my_list.selectedIndices.length;
for (var i:Number = 0; i < numSelected; i++) {
  trace("selectedIndices[" + i + "] = "+
  my_list.getItemAt(my_list.selectedIndices[i]).label);
}
```

Voir aussi

`List.selectedIndex`, `List.selectedItem`, `List.selectedItems`

List.selectedItem

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.selectedItem

Description

Propriété (lecture seule), objet d'élément dans une liste à sélection unique. (Dans une liste à sélection multiple où plusieurs éléments sont sélectionnés, `selectedItem` renvoie l'élément qui a été sélectionné le plus récemment). S'il n'y a aucune sélection, la valeur est `undefined`.

Exemple

Cet exemple affiche l'étiquette sélectionnée :

```
trace(my_list.selectedItem.label);
```

L'exemple suivant affiche le contenu d'un élément sélectionné chaque fois que l'utilisateur sélectionne un nouvel élément dans la liste :

```
my_list.multipleSelection = true;

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

// Création d'un objet écouteur.
var listListener:Object = new Object();
listListener.change = function(evt_obj:Object) {
  // Affichage de chaque propriété de l'objet
  var tempStr:String = "[object";
  for (var i:String in evt_obj.target.selectedItem) {
    tempStr += " " + i + ":'" + evt_obj.target.selectedItem[i]+"'";
  }
  tempStr += "]";
  trace(tempStr);
};
// Ajout de l'écouteur.
my_list.addEventListener("change", listListener);
```

Voir aussi

`List.selectedIndex`, `List.selectedIndices`, `List.selectedItems`

List.selectedItems

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.selectedItems

Description

Propriété (lecture seule), tableau des objets d'élément sélectionnés. Dans une liste à sélection multiple, `selectedItems` vous permet d'accéder au jeu d'éléments sélectionnés comme objets d'éléments.

Exemple

L'exemple suivant récupère un tableau des objets d'éléments sélectionnés :

```
var myObjArray:Array = my_list.selectedItems;
```

L'exemple suivant affiche deux occurrences du composant List sur la scène. Lorsqu'un utilisateur sélectionne un élément dans la première liste, l'élément sélectionné est copié dans la deuxième liste. Pour tester ce code, vous devez ajouter une copie du composant List dans la bibliothèque du document en cours. Ajoutez le code suivant à l'image 1 du scénario :

```
this.createClassObject(mx.controls.List, "my_list", 10,
  {multipleSelection:true});
my_list.setSize(200, 100);

this.createClassObject(mx.controls.List, "selectedItems_list", 20,
  {selectable:false});
selectedItems_list.setSize(200, 100);
selectedItems_list.move(0, 110);

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

var listListener:Object = new Object();
listListener.change = function(evt_obj:Object) {
  trace("You have selected " + my_list.selectedItems.length + " items.");
  selectedItems_list.dataProvider = my_list.selectedItems;
}
my_list.addEventListener("change", listListener);
```

Voir aussi

`List.selectedIndex`, `List.selectedItem`, `List.selectedIndices`

List.setPropertiesAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.setPropertiesAt(*index*, *styleObj*)

Paramètres

index Nombre supérieur à 0 ou inférieur à `List.length` indiquant l'index de l'élément à modifier.

styleObj Objet énumérant les propriétés et valeurs à définir.

Valeur renvoyée

Aucune.

Description

Méthode qui applique les propriétés spécifiées à l'élément donné. Les propriétés prises en charge sont `icon` et `backgroundColor`.

Exemple

L'exemple suivant modifie la couleur d'arrière-plan du troisième élément sur le rouge et lui attribue une icône. Pour tester ce code, faites glisser un composant List sur la scène et nommez l'occurrence **my_list**. Ajoutez ensuite un symbole MovieClip/Graphic à la bibliothèque avec un identificateur de liaison « file ». Ajoutez le code suivant à l'image 1 du scénario :

```
var my_list:mx.controls.List;

my_list.setSize(200, 100);

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

my_list.setPropertiesAt(2, {backgroundColor:0xFF0000, icon: "file"});
```

List.sortItems()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
listInstance.sortItems(compareFunc)
```

Paramètres

compareFunc Référence à une fonction. Cette fonction est utilisée pour comparer deux éléments afin de déterminer leur ordre de tri.

Pour plus d'informations, reportez-vous à la méthode `Array.sort()` dans le *Guide de référence du langage ActionScript 2.0*.

Valeur renvoyée

Aucune.

Description

Méthode : trie les éléments de la liste en fonction du paramètre *compareFunc*.

Exemple

Les exemples suivants trient les éléments en fonction de leurs étiquettes en majuscules. Notez que les paramètres `a` et `b` transmis à la fonction sont des éléments qui ont des propriétés `label` et `data`.

```
var my_list:mx.controls.List;

my_list.setSize(200, 100);
my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

my_list.sortItems(upperCaseFunc);
function upperCaseFunc(a:Object, b:Object):Boolean {
    return (a.label.toUpperCase() > b.label.toUpperCase());
}
```

Voir aussi

`List.sortItemsBy()`

List.sortItemsBy()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
listInstance.sortItemsBy(fieldName, optionsFlag)
listInstance.sortItemsBy(fieldName, order)
```

Paramètres

fieldName Chaîne spécifiant le nom du champ à utiliser pour le tri. Cette valeur est en général « `label` » ou « `data` ».

order Chaîne qui indique si le tri des éléments doit être effectué dans l'ordre croissant (`"ASC"`) ou décroissant (`"DESC"`).

optionsFlag Permet d'effectuer plusieurs types de tris dans un même tableau sans avoir à copier ce dernier ou à le trier à plusieurs reprises.

Les valeurs possibles pour *optionsFlag* sont les suivantes :

- `Array.DESCENDING` trie par ordre décroissant.
- `Array.CASEINSENSITIVE` trie sans respecter la casse.
- `Array.NUMERIC` trie par ordre numérique si les deux éléments comparés sont des nombres. S'il ne s'agit pas de nombres, effectuez une comparaison de type chaîne (qui peut être non sensible à la casse si l'indicateur est spécifié).
- `Array.UNIQUESORT` renvoie un code d'erreur (0) au lieu d'un tableau trié si deux objets sont identiques ou comportent des champs de tri identiques.
- `Array.RETURNINDEXEDARRAY` renvoie un tableau d'index de nombres entiers correspondant au résultat du tri. Par exemple, le tableau suivant renverra la seconde ligne de code et ne sera pas modifié :

```
["a", "d", "c", "b"]
[0, 3, 2, 1]
```

Vous pouvez combiner ces options en une seule valeur. Par exemple, le code suivant associe les options 3 et 1 :

```
my_array.sort (Array.NUMERIC | Array.DESCENDING)
```

Valeur renvoyée

Aucune.

Description

Méthode qui trie les éléments de la liste dans l'ordre spécifié, via le nom de champ spécifié. Si les éléments *fieldName* sont une combinaison de chaînes de texte et de nombres entiers, ce sont les éléments entiers qui sont indiqués en premier. Le paramètre *fieldName* est généralement « `label` » ou « `data` », mais vous pouvez spécifier n'importe quelle primitive.

Cette méthode représente le moyen le plus rapide pour trier les données dans un composant. Elle permet également de conserver l'état de sélection du composant. La méthode `sortItemsBy()` est rapide, car elle n'exécute pas de code ActionScript pendant le tri. La méthode `sortItems()` doit exécuter une fonction de comparaison ActionScript et se révèle donc plus lente.

Exemple

Le code suivant trie les éléments de la liste par ordre croissant en utilisant leurs étiquettes.

```
var my_list:mx.controls.List;

my_list.setSize(200, 100);
my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});

my_list.sortItemsBy("label", "ASC");
```

Voir aussi

`List.sortItems()`

List.vPosition

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.vPosition

Description

Propriété qui définit le premier élément visible de la liste. Si vous donnez à cette propriété un numéro d'index qui n'existe pas, la liste défile jusqu'à l'index le plus proche. La valeur par défaut est 0.

Exemple

L'exemple suivant affiche la valeur actuelle de la vPosition de la liste chaque fois que vous faites défiler son contenu :

```
my_list.setSize(200, 60);
my_list.rowCount = 4;
my_list.vPosition = 2;

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"flex", label:"Flex"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"fireworks", label:"Fireworks"});
my_list.addItem({data:"contribute", label:"Contribute"});
my_list.addItem({data:"breeze", label:"Breeze"});

var listListener:Object = new Object();
listListener.scroll = function(evt_obj:Object) {
  trace("my_list.vPosition = " + my_list.vPosition);
}
my_list.addEventListener("scroll", listListener);
```

List.vScrollPolicy

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

listInstance.vScrollPolicy

Description

Propriété, chaîne déterminant si la liste gère le défilement vertical. La valeur de cette propriété peut être "on", "off" ou "auto". La valeur "auto" fait apparaître une barre de défilement lorsque cela est nécessaire.

Exemple

L'exemple suivant désactive la barre de défilement vertical pour une liste :

```
var my_list:mx.controls.List;

my_list.setSize(200, 60);
my_list.rowCount = 4;
my_list.vScrollPolicy = "off";

my_list.addItem({data:"flash", label:"Flash"});
my_list.addItem({data:"flex", label:"Flex"});
my_list.addItem({data:"coldfusion", label:"ColdFusion"});
my_list.addItem({data:"dreamweaver", label:"Dreamweaver"});
my_list.addItem({data:"fireworks", label:"Fireworks"});
my_list.addItem({data:"contribute", label:"Contribute"});
my_list.addItem({data:"breeze", label:"Breeze"});

var listListener:Object = new Object();
listListener.scroll = function(evt_obj:Object) {
  trace("my_list.vPosition = " + my_list.vPosition);
}
my_list.addEventListener("scroll", listListener);
```

Vous pouvez toujours créer un défilement en utilisant `List.vPosition`, la souris ou le clavier.

Voir aussi

`List.vPosition`

CHAPITRE 28

Composant Loader

28

Le composant Loader est un conteneur qui peut afficher un fichier SWF ou JPEG (mais pas les fichiers JPEG *progressifs*). Vous pouvez redimensionner le contenu du chargeur ou le chargeur lui-même pour l'adapter à la taille du contenu. Par défaut, le contenu est dimensionné pour s'ajuster au chargeur. Vous pouvez également charger du contenu à l'exécution et surveiller la progression du chargement (même si une fois que le contenu est chargé, il est mis en mémoire cache et la progression passe donc rapidement à 100 %).

REMARQUE

Le composant Loader est pris en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Un composant Loader ne peut recevoir le focus. Cependant, le contenu chargé dans le composant Loader peut accepter le focus et avoir ses propres interactions de focus. Pour plus d'informations sur le contrôle du focus, reportez-vous à « Classe FocusManager », à la page 745 ou à « Création de la navigation personnalisée du focus » dans *Utilisation des composants ActionScript 2.0*.

Un aperçu en direct de chaque occurrence de Loader reflète les modifications effectuées sur les paramètres dans l'inspecteur Propriétés ou l'inspecteur des composants pendant la programmation.

Vous pouvez utiliser le panneau Accessibilité pour faire en sorte que les logiciels de lecture d'écran puissent accéder au contenu du composant Loader. Pour plus d'informations, reportez-vous à *Création de contenu accessible* dans *Utilisation de Flash*.

Utilisation du composant Loader

Vous pouvez utiliser un chargeur chaque fois que vous devez récupérer du contenu depuis un emplacement distant et le placer dans une application Flash. Par exemple, vous pouvez utiliser un chargeur pour ajouter un logo d'entreprise (fichier JPEG) dans un formulaire. Vous pouvez également utiliser un chargeur pour exploiter un travail Flash qui a déjà été terminé.
Par exemple, si vous avez déjà construit une application Flash et que vous souhaitez l'étendre, vous pouvez utiliser le chargeur pour placer l'ancienne application dans une nouvelle application, éventuellement comme section d'une interface d'onglets. Dans un autre exemple, vous pouvez utiliser le composant Loader dans une application qui affiche des photos. Utilisez `Loader.load()`, `Loader.percentLoaded` et `Loader.complete` pour contrôler la synchronisation des chargements d'images et afficher des barres de progression pour l'utilisateur lors de ces opérations.

Si vous chargez certains composants dans un fichier SWF ou dans le composant Loader, il se peut qu'ils ne fonctionnent pas correctement. Ces composant comprennent : Alert, ComboBox, DateField, Menu, MenuBar et Window.

Lors d'un appel à la méthode `loadMovie()` ou d'un chargement dans le composant Loader, utilisez la propriété `_lockroot`. Si vous utilisez le composant Loader, ajoutez le code suivant :

```
myLoaderComponent.content._lockroot = true;
```

Si vous utilisez un clip en appelant la méthode `loadMovie()`, ajoutez le code suivant :

```
myMovieClip._lockroot = true;
```

Si vous ne définissez pas `_lockroot` sur `true` dans le clip du chargeur, ce dernier peut accéder uniquement à sa propre bibliothèque, et non à celle du clip chargé.

Flash Player 7 prend en charge la propriété `_lockroot`. Pour plus d'informations sur cette propriété, reportez-vous à la propriété `MovieClip._lockroot` dans le *Guide de référence du langage ActionScript 2.0*.

Les composants tels que Loader, ScrollPane et Window ont des événements pour déterminer à quel moment le chargement du contenu est terminé. Ainsi, si vous souhaitez définir des propriétés sur le contenu d'un composant Loader, ScrollPane ou Window, ajoutez l'instruction de propriété dans un gestionnaire d'événements « complete », comme indiqué dans l'exemple suivant :

```
loadtest = new Object();
loadtest.complete = function(eventObject){
  content_mc._rotation= 45;
}
my_loader.addEventListener("complete", loadtest)
```

Pour plus d'informations, voir la section « Loader.complete », à la page 856.

Paramètres du composant Loader

Dans l'inspecteur des propriétés ou l'inspecteur de composants (Fenêtre > Inspecteur de composants), vous pouvez définir les paramètres de création suivants pour chaque occurrence Loader :

autoload indique si le contenu doit être chargé automatiquement (`true`) ou s'il faut attendre jusqu'à ce que la méthode `Loader.load()` soit appelée (`false`). The default value is `true`.

contentPath URL absolue ou relative indiquant le fichier à charger dans le chargeur. Un chemin relatif doit être relatif au fichier SWF chargeant le contenu. L'URL doit se trouver dans le même sous-domaine que l'URL où se trouve le contenu Flash. Pour une utilisation dans Flash Player ou en mode animation, tous les fichiers SWF doivent être stockés dans un même dossier et les noms de fichiers ne doivent pas inclure de spécifications de dossier ni de disque. La valeur par défaut est `undefined` jusqu'à ce que le chargement commence.

Le composant Loader peut charger le contenu d'autres domaines, *si* vous possédez des fichiers de régulation dans ces domaines. Reportez-vous à « Autorisation de chargement de données inter-domaines » dans le guide *Formation à ActionScript 2.0 dans Adobe Flash*.

scaleContent indique si le contenu est redimensionné pour s'adapter au chargeur (`true`) ou si le chargeur est redimensionné pour s'adapter au contenu (`false`). The default value is `true`.

Dans l'inspecteur des composants, vous pouvez définir les paramètres supplémentaires suivants pour chaque occurrence du composant Loader (Fenêtre > Inspecteur de composants) :

enabled est une valeur booléenne qui indique si le composant peut recevoir le focus et l'entrée. The default value is `true`.

visible est une valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`). The default value is `true`.

REMARQUE

Les propriétés minHeight et minWidth sont utilisées par des routines de dimensionnement internes. Elles sont définies dans UIObject et remplacées par différents composants selon vos besoins. Ces propriétés peuvent être utilisées pour créer un gestionnaire de dispositions personnalisées. Dans le cas contraire, la définition de ces propriétés dans l'inspecteur de composants n'a aucun effet visible.

ActionScript vous permet de définir des options supplémentaires pour les occurrences de Loader en utilisant ses méthodes, propriétés et événements. Pour plus d'informations, voir « Classe Loader », à la page 849.

Création d'une application avec le composant Loader

La procédure suivante explique comment ajouter un composant Loader à une application pendant la programmation. Dans cet exemple, le chargeur charge un logo au format JPEG depuis une URL imaginaire.

Pour créer une application avec le composant Loader :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser un composant Loader du panneau Composants jusqu'à la scène.
3. Dans l'inspecteur Propriétés, entrez le nom d'occurrence **flower**.
4. Sélectionnez le chargeur (composant Loader) sur la scène et dans l'inspecteur des propriétés, puis entrez **http://www.helpexamples.com/flash/images/image1.jpg** pour le paramètre contentPath.
5. Choisissez Contrôle > Tester l'animation.

Pour créer une occurrence du composant Loader à l'aide d'ActionScript :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser le composant Loader du panneau Composants jusqu'à la bibliothèque.
3. Sélectionnez la première image dans le scénario principal, ouvrez le panneau Actions et indiquez le code suivant :

```
this.createClassObject(mx.controls.Loader, "my_loader", 1);
my_loader.contentPath = "http://www.helpexamples.com/flash/images/
  image1.jpg";
```

   Ce script utilise la méthode « UIObject.createClassObject() », à la page 1416 pour créer l'occurrence du composant Loader.
4. Sélectionnez Contrôle > Tester l'animation.

Personnalisation du composant Loader

Vous pouvez transformer un composant Loader de façon horizontale et verticale pendant la programmation et lors de l'exécution. Lors de la programmation, sélectionnez le composant sur la scène et utilisez l'outil Transformer librement ou une commande Modification > Transformer. A l'exécution, utilisez la méthode `setSize()` (voir `UIObject.setSize()`).

Le comportement de dimensionnement du composant Loader est contrôlé par la propriété `scaleContent`. Lorsque `scaleContent` est défini sur `true`, le contenu est dimensionné afin de rester dans les limites du chargeur (et redimensionné lorsque `UIObject.setSize()` est appelée). Lorsque `scaleContent` est défini sur `false`, la taille du composant est fixée sur celle du contenu et `UIObject.setSize()` n'a aucun effet.

Utilisation de styles avec le composant Loader

Le composant Loader utilise les styles suivants.

Style	Thème	Description
`borderStyle`	Les deux	Le composant Loader utilise une occurrence de RectBorder comme bordure et répond aux styles définis sur cette classe. Voir « Classe RectBorder », à la page 1111. Le style de bordure par défaut est `"none"`.

Exemple :

```
my_ldr.setStyle("backgroundColor", 0xEEEEEE);
```

Pour plus d'informations, reportez-vous à « Utilisation de styles pour personnaliser la couleur et le texte des composants » dans *Utilisation des composants ActionScript 2.0*.

Utilisation d'enveloppes avec le composant Loader

Le composant Loader utilise une occurrence de RectBorder pour sa bordure (voir « Classe RectBorder », à la page 1111).

Classe Loader

Héritage MovieClip > Classe UIObject > Classe UIComponent > View > Loader

Nom de classe ActionScript mx.controls.Loader

Les propriétés de la classe Loader vous permettent de définir le contenu de sorte qu'il se charge et contrôle sa propre progression de chargement lors de l'exécution.

La définition d'une propriété de la classe Loader avec ActionScript annule le paramètre du même nom défini dans l'inspecteur Propriétés ou l'inspecteur des composants.

Toutes les classes de composants ont une propriété `version` qui correspond à une propriété de classe. Les propriétés de classe sont uniquement disponibles sur la classe elle-même.
La propriété `version` renvoie une chaîne qui indique la version du composant. Pour accéder à cette propriété, utilisez le code suivant :

```
trace(mx.controls.Loader.version);
```

REMARQUE

Le code `trace(myLoaderInstance.version);` renvoie `undefined`.

Méthodes de la classe Loader

Le tableau suivant présente la méthode de la classe Loader.

Méthode	Description
`Loader.load()`	Charge le contenu spécifié par la propriété `contentPath`.

Méthodes héritées de la classe UIObject

Le tableau suivant répertorie les méthodes de la classe Loader héritées de la classe UIObject. Pour appeler ces méthodes à partir de l'objet Loader, utilisez le formulaire *`LoaderInstance.methodName`*.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet sur la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet sur un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque des paramètres ont été définis dans les inspecteurs de propriétés et de composants.
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image courante.
`UIObject.setSize()`	Redimensionne l'objet aux dimensions demandées.

Méthode	Description
UIObject.setSkin()	Définit une enveloppe dans l'objet.
UIObject.setStyle()	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant répertorie les méthodes de la classe Loader héritées de la classe UIComponent. Pour appeler ces méthodes à partir de l'objet Loader, utilisez le formulaire *LoaderInstance.methodName*.

Méthode	Description
UIComponent.getFocus()	Renvoie une référence à l'objet ayant le focus.
UIComponent.setFocus()	Attribue le focus à l'occurrence de composant.

Propriétés de la classe Loader

Le tableau suivant répertorie les propriétés de la classe Loader.

Propriété	Description
Loader.autoLoad	Valeur booléenne indiquant si le contenu se charge automatiquement (true) ou si vous devez appeler Loader.load() (false).
Loader.bytesLoaded	Propriété en lecture seule indiquant le nombre d'octets ayant été chargés.
Loader.bytesTotal	Propriété en lecture seule indiquant le nombres d'octets du contenu.
Loader.content	Référence au contenu du chargeur. Cette propriété est en lecture seule.
Loader.contentPath	Chaîne indiquant l'URL du contenu à charger.
Loader.percentLoaded	Nombre indiquant le pourcentage de contenu chargé. Cette propriété est en lecture seule.
Loader.scaleContent	Valeur booléenne indiquant si le contenu est dimensionné pour s'ajuster au chargeur (true) ou si le chargeur est dimensionné pour s'ajuster au contenu (false).

Propriétés héritées de la classe UIObject

Le tableau suivant répertorie les propriétés de la classe Loader héritées de la classe UIObject. Pour accéder à ces propriétés à partir de l'objet Loader, utilisez le formulaire *`LoaderInstance.propertyName`*.

Propriété	Description
`UIObject.bottom`	Position du bord inférieur de l'objet par rapport au bord inférieur de son parent. Lecture seule.
`UIObject.height`	Hauteur de l'objet, en pixels. Lecture seule.
`UIObject.left`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.right`	Position du bord droit de l'objet par rapport au bord droit de son parent. Lecture seule.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Position du bord supérieur de l'objet par rapport à son parent. Lecture seule.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Largeur de l'objet, en pixels. Lecture seule.
`UIObject.x`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.y`	Bord supérieur de l'objet, en pixels. Lecture seule.

Propriétés héritées de la classe UIComponent

Le tableau suivant répertorie les propriétés de la classe Loader héritées de la classe UIComponent. Pour accéder à ces propriétés à partir de l'objet Loader, utilisez le formulaire *`LoaderInstance.propertyName`*.

Propriété	Description
`UIComponent.enabled`	Indique si le composant peut recevoir le focus et la saisie.
`UIComponent.tabIndex`	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Evénements de la classe Loader

Le tableau suivant répertorie les événements de la classe Loader.

Evénement	Description
`Loader.complete`	Déclenché à la fin du chargement du contenu.
`Loader.progress`	Déclenché pendant le chargement du contenu.

Evénements hérités de la classe UIObject

Le tableau suivant répertorie les événements de la classe Loader hérités de la classe UIObject.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant répertorie les événements de la classe Loader hérités de la classe UIComponent.

Evénement	Description
`UIComponent.focusIn`	Diffusé lorsqu'un objet reçoit le focus.
`UIComponent.focusOut`	Diffusé lorsqu'un objet perd le focus.
`UIComponent.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIComponent.keyUp`	Diffusé lorsqu'une touche est relâchée.

Loader.autoLoad

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

loaderInstance.autoLoad

Description

Propriété : valeur booléenne indiquant s'il faut charger le contenu automatiquement (`true`) ou attendre jusqu'à ce que `Loader.load()` soit appelée (`false`). The default value is `true`.

Exemple

Le code suivant configure le composant Loader pour qu'il attende un appel `Loader.load()` :

```
loader.autoload = false;
```

Loader.bytesLoaded

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

loaderInstance.bytesLoaded

Description

Propriété (lecture seule) : nombre d'octets de contenu ayant été chargés. La valeur par défaut est 0 jusqu'à ce que le chargement commence.

Exemple

Lorsqu'un composant Loader et un composant ProgressBar se trouvent dans la bibliothèque du document actif, le code suivant crée des occurrences de chargeur et de barre de progression. Il crée ensuite un objet écouteur avec un gestionnaire d'événements `progress` qui affiche la progression du chargement. L'écouteur est enregistré avec l'occurrence `my_ldr`.

Lorsque vous créez une occurrence avec `createClassObject()`, vous devez la placer sur la scène à l'aide de `move()`. Voir `UIObject.move()`.

```
import mx.controls.Loader;
import mx.controls.ProgressBar;

System.security.allowDomain("http://www.flash-mx.com");

this.createClassObject(Loader, "my_ldr", 10);
this.createClassObject(ProgressBar, "my_pb", 20, {source:"my_ldr"});

my_ldr.move(1, 50);
my_pb.move(1, 1);

var loaderListener:Object = new Object();
loaderListener.progress = function(evt_obj:Object) {
 // evt_obj.target est le composant qui a généré l'événement progress,
 // c'est-à-dire le chargeur.
 my_pb.setProgress(my_ldr.bytesLoaded, my_ldr.bytesTotal);
 // Affichage de la progression.
};
my_ldr.addEventListener("progress", loaderListener);
my_ldr.contentPath = "http://www.flash-mx.com/images/image2.jpg";
```

Loader.bytesTotal

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`loaderInstance``.bytesTotal`

Description

Propriété (lecture seule) : taille du contenu en octets. La valeur par défaut est 0 jusqu'à ce que le chargement commence.

Exemple

Le code suivant crée une barre de progression et un composant Loader. Il crée ensuite un objet écouteur de chargement avec un gestionnaire d'événements `progress` qui affiche la progression du chargement. L'écouteur est enregistré avec l'occurrence `my_ldr` de la manière suivante :

```
import mx.controls.Loader;
import mx.controls.ProgressBar;
this.createClassObject(ProgressBar, "my_pb", 998);
this.createClassObject(Loader, "my_ldr", 999);
my_pb.move(1, 1);
my_ldr.move(1, 50);
my_pb.source = "my_ldr";
var loadListener:Object = new Object();
loadListener.progress = function(eventObj){
  // eventObj.target est le composant qui a généré l'événement progress,
  // c'est-à-dire le chargeur.
  my_pb.setProgress(my_ldr.bytesLoaded, my_ldr.bytesTotal);
    // Affichage de la progression.
}
my_ldr.addEventListener("progress", loadListener);
my_ldr.contentPath = "http://www.flash-mx.com/images/image2.jpg";
```

Voir aussi

`Loader.bytesLoaded`

Loader.complete

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.complete = function(eventObj:Object){
  // ...
};
loaderInstance.addEventListener("complete", listenerObject);
```

Utilisation 2 :

```
on (complete) {
  // ...
}
```

Description

Evénement : diffusé à tous les écouteurs enregistrés une fois le contenu chargé.

Le premier exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*loaderInstance*) distribue un événement (ici, `complete`) qui est géré par une fonction, également appelée un *gestionnaire*, associée à un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence Loader. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de composant. Par exemple, le code suivant, associé à l'occurrence Loader `myLoaderComponent`, envoie « _level0.myLoaderComponent » vers le panneau Sortie :

```
on (complete) {
  trace(this);
}
```

Exemple

L'exemple suivant crée un composant Loader, `my_ldr`, puis définit un objet écouteur pour un événement `complete`. L'exemple charge une image à partir d'une page Web. A la fin du chargement, l'écouteur affiche un message dans le panneau Sortie.

Faites glisser un composant Loader jusqu'à la bibliothèque, puis ajoutez le code suivant dans la première image du scénario.

```
/**
 Requiert :
  - Composant Loader dans la bibliothèque.
*/

System.security.allowDomain("http://www.flash-mx.com");

// Création de l'occurrence Loader.
this.createClassObject(mx.controls.Loader, "my_ldr", 10);
```

```
// Création d'un objet écouteur.
var loaderListener:Object = new Object();
loaderListener.complete = function(evt_obj:Object){
   trace("loading complete");
}

// Ajout de l'écouteur.
my_ldr.addEventListener("complete", loaderListener);
my_ldr.load("http://www.flash-mx.com/images/image2.jpg");
```

Loader.content

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

loaderInstance.content

Description

Propriété (lecture seule) : référence à une occurrence de clip comportant le contenu du fichier chargé. La valeur est `undefined` jusqu'à ce que le chargement commence. Définissez des propriétés pour le contenu avec une fonction de gestionnaire d'événements pour l'événement `Loader.complete`.

Exemple

Le composant Loader comporte un événement « complete » pour que vous puissiez vous assurer que le contenu est complètement chargé avant d'accéder aux propriétés de son contenu.

L'exemple suivant utilise la propriété `Loader.content` dans une fonction de gestionnaire d'événements pour l'événement « complete ». Faites glisser un composant Loader du panneau Composants vers la bibliothèque du document actif pour qu'il y apparaisse. Ajoutez ensuite le code ActionScript suivant à la première image du scénario principal :

```
this.createClassObject(mx.controls.Loader, "my_ldr", 10);
my_ldr.contentPath = "http://www.flash-mx.com/images/image1.jpg";
// Affectation d'une variable au contenu.
var content_mc:MovieClip = my_ldr.content;

var loadtest:Object = new Object();
loadtest.complete = function(){
```

```
// Définition des propriétés pour le contenu.
  content_mc._alpha = 50;
  content_mc._rotation= 45;
  trace(content_mc._width);
}
my_ldr.addEventListener("complete", loadtest);
```

Loader.contentPath

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

loaderInstance.contentPath

Description

Propriété : chaîne indiquant l'URL absolue ou relative du fichier à charger dans le chargeur. Un chemin relatif doit être relatif au fichier SWF chargeant le contenu. L'URL doit se trouver dans le même sous-domaine que le fichier SWF en chargement.

Si vous utilisez Flash Player ou le mode animation dans Flash, tous les fichiers SWF doivent être stockés dans le même dossier et les noms de fichiers ne doivent pas inclure d'informations de dossier ni de disque.

Exemple

L'exemple suivant demande à l'occurrence de chargeur d'afficher le contenu du fichier logo.swf :

```
flower.contentPath = "http://www.flash-mx.com/images/image1.jpg"
```

L'exemple suivant décharge le contenu du composant Loader lorsque vous cliquez sur l'occurrence Button `my_btn` :

```
flower.contentPath = "http://www.flash-mx.com/images/image1.jpg"
function clicked(){
  flower.contentPath = "";
}
my_btn.addEventListener("click", clicked);
```

Loader.load()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

loaderInstance.load([*path*])

Paramètres

path Paramètre facultatif qui indique la valeur de la propriété contentPath avant le début du chargement. Si aucune valeur n'est spécifiée, la valeur actuelle de contentPath est utilisée telle quelle.

Valeur renvoyée

Aucune.

Description

Méthode : demande au chargeur de commencer le chargement de son contenu.

Exemple

L'exemple suivant crée une occurrence du composant Loader, my_ldr, et une occurrence du composant Button, puis définit la propriété du chargeur autoload sur false pour que le chargement ne commence pas tant que la méthode load() n'a pas été appelée. L'exemple définit ensuite contentPath à l'emplacement Web d'une image et crée un écouteur pour un événement click sur le bouton. Lorsque l'utilisateur clique sur le bouton, le gestionnaire d'événements appelle my_ldr.load() pour charger l'image. Le gestionnaire d'événements désactive également le bouton.

Faites glisser un composant Loader et un composant Button du panneau Composant vers la bibliothèque, puis ajoutez le code suivant à la première image du scénario.

```
/**
 Requiert :
  - Composant Loader dans la bibliothèque.
  - composant Button dans la bibliothèque.
*/

System.security.allowDomain("http://www.flash-mx.com");
```

```
// Création de l'occurrence Loader.
this.createClassObject(mx.controls.Loader, "my_ldr", 10);
this.createClassObject(mx.controls.Button, "load_button", 20, {label:"Load
  image"});

my_ldr.move(0, 30);

my_ldr.autoLoad = false;
my_ldr.contentPath = "http://www.flash-mx.com/images/image1.jpg";

var loadListener:Object = new Object();
loadListener.click = function (evt_obj:Object) {
 my_ldr.load();
 load_button.enabled = false;
}
load_button.addEventListener("click", loadListener);
```

Loader.percentLoaded

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

loaderInstance.percentLoaded

Description

Propriété (lecture seule) : nombre indiquant le pourcentage de contenu ayant été chargé. Cette propriété est généralement utilisée pour présenter la progression du chargement à l'utilisateur de façon plus claire. Utilisez le code suivant pour arrondir le chiffre au nombre entier le plus proche :

```
Math.round(bytesLoaded/bytesTotal*100))
```

Exemple

L'exemple suivant crée une occurrence Loader, puis un objet écouteur avec un gestionnaire `progress` qui suit le pourcentage chargé et l'envoie vers le panneau Sortie :

```
import mx.controls.Loader;
this.createClassObject(Loader, "my_ldr", 999);
var loadListener:Object = new Object();
loadListener.progress = function(eventObj) {
  // eventObj.target est le composant qui a généré l'événement progress,
  // c'est-à-dire le chargeur.
  trace("The image is "+my_ldr.percentLoaded+"% loaded.");
  // Suivi de la progression du chargement.
};
my_ldr.addEventListener("progress", loadListener);
my_ldr.contentPath = "http://www.flash-mx.com/images/image2.jpg";
```

Loader.progress

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.progress = function(eventObj:Object) {
  // ...
};
loaderInstance.addEventListener("progress", listenerObject);
```

Utilisation 2 :

```
on (progress) {
  // ...
}
```

Description

Evénement : diffusé à l'ensemble des écouteurs enregistrés pendant le chargement du contenu. Cet événement se produit lorsque le chargement est déclenché par le paramètre `autoload` ou par un appel à `Loader.load()`. L'événement progress n'est pas toujours diffusé, et l'événement `complete` peut être diffusé sans qu'aucun événement `progress` ne soit distribué. Ceci peut se produire si le contenu chargé est un fichier local.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence de composant (*loaderInstance*) distribue un événement (ici, progress) qui est géré par une fonction, également appelée un *gestionnaire*, associée à un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode EventDispatcher.addEventListener() sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel à un gestionnaire on() et doit être associé directement à une occurrence Loader. Le mot-clé this, utilisé dans un gestionnaire on() lié à un composant, correspond à l'occurrence de composant. Par exemple, le code suivant, associé à l'occurrence Loader myLoaderComponent, envoie « _level0.myLoaderComponent » vers le panneau Sortie :

```
on (progress) {
  trace(this);
}
```

Exemple

Le code suivant crée une occurrence Loader, puis un objet écouteur avec un gestionnaire d'événements pour l'événement progress. Celui-ci envoie un message vers le panneau Sortie indiquant le pourcentage du contenu qui a été chargé :

```
//Création de l'occurence Loader.
this.createClassObject(mx.controls.Loader, "my_ldr", 10);

// Création d'un objet écouteur.
var loaderListener:Object = new Object();
loaderListener.progress = function(evt_obj:Object){
  // evt_obj.target est le composant qui a généré l'événement progress,
  // c'est-à-dire le chargeur.
  trace("image is " + my_ldr.percentLoaded + "% loaded.");
}

// Ajout de l'écouteur.
my_ldr.addEventListener("progress", loaderListener);

// Affectation du chemin du contenu du chargeur.
my_ldr.contentPath = "http://www.flash-mx.com/images/image1.jpg";
```

Loader.scaleContent

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

loaderInstance.scaleContent

Description

Propriété : indique si la taille du contenu s'ajuste pour correspondre au chargeur (true), ou si le chargeur s'ajuste pour correspondre au contenu (`false`). The default value is `true`.

Exemple

Le code suivant demande au chargeur de se redimensionner en fonction de la taille de son contenu :

```
my_ldr.scaleContent = false;
```

CHAPITRE 29

Composants média

29

Les composants média en flux continu facilitent l'incorporation de supports en flux continu dans les diaporamas Adobe Flash. Ces composants vous permettent de présenter vos supports de différentes manières.

REMARQUE

Les composants média sont pris en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Vous pouvez utiliser les trois composant médias suivants :

- Le composant MediaDisplay permet de diffuser un support en flux continu dans votre contenu Flash sans avoir recours à une interface utilisateur. Vous pouvez utiliser ce composant avec des données vidéo et audio. Lorsque vous l'utilisez seul, vous n'avez aucun contrôle sur le support.
- Le composant MediaController fournit des contrôles d'interface utilisateur standard (lecture, pause, etc.) pour la lecture du support. Les supports ne sont jamais chargés dans ce composant ni lus par ce dernier. Il est uniquement utilisé pour contrôler la lecture dans une occurrence de MediaPlayback ou de MediaDisplay. Le composant MediaController comporte un « tiroir » qui affiche le contenu des contrôles de lecture lorsque la souris est positionnée sur le composant.
- Le composant MediaPlayback est une combinaison des composants MediaDisplay et MediaController. Il fournit des méthodes permettant de lire en continu votre contenu multimédia.

N'oubliez pas les points suivants concernant les composants média :

- Les composants média requièrent Flash Player 6 ou une version ultérieure. Dans Flash Player 6, les composants média prennent en charge les fichiers FLV uniquement par le biais de Flash Communication Server, pas de HTTP.
- Les composants média ne prennent pas en charge la fonctionnalité de défilement vers l'avant et l'arrière. Vous pouvez néanmoins arriver au même résultat en déplaçant le curseur de lecture.
- Seules la taille du composant et la politique du contrôleur sont reflétées dans l'aperçu en direct.
- Les composants média ne prennent pas en charge l'accessibilité.

Interaction avec les composants média

Les composants en flux continu MediaPlayback et MediaController répondent aux interactions avec le clavier et la souris, contrairement au composant MediaDisplay. Le tableau suivant résume les actions des composants MediaPlayback et MediaController lorsqu'ils reçoivent le focus.

Cible	Navigation	Description
Contrôles de lecture d'un contrôleur donné	Survol de la souris	Le bouton est mis en évidence.
Contrôles de lecture d'un contrôleur donné	Clic unique sur le bouton gauche de la souris	Les utilisateurs peuvent cliquer sur les contrôles de lecture pour manipuler la lecture des supports vidéo et audio. Les boutons Pause/Lire et Rembobiner/Atteindre la fin suivent les comportements standard des boutons. Lorsque l'utilisateur appuie sur le bouton de la souris, le bouton à l'écran apparaît dans son état enfoncé. Lorsque l'utilisateur relâche le bouton de la souris, le bouton reprend son aspect non sélectionné. Le bouton Atteindre la fin est désactivé lors de la lecture des fichiers de support FLV.

Cible	Navigation	Description
Contrôles de curseur d'un contrôleur donné	Déplacement du curseur vers l'avant et vers l'arrière	La barre de lecture indique la position de l'utilisateur dans le support. Le curseur de lecture se déplace horizontalement (par défaut) pour indiquer la lecture du début (à gauche) jusqu'à la fin (à droite). Le curseur se déplace de bas en haut lorsque les contrôles sont orientés à la verticale. Au fur et à mesure que le curseur se déplace de la gauche vers la droite, il met en évidence l'espace d'affichage précédent pour indiquer que ce contenu a été lu ou sélectionné. L'espace d'affichage devant le curseur n'est pas mis en évidence jusqu'au passage du curseur. Les utilisateurs peuvent faire glisser la poignée de l'indicateur pour modifier la position de lecture du contenu multimédia. Si le support est en cours de lecture, la lecture commence automatiquement à l'endroit où l'utilisateur relâche le bouton de la souris. Si le support est en pause, le curseur peut être déplacé puis relâché et le support reste en pause. Vous pouvez également utiliser un curseur de volume, qui peut être déplacé de gauche (muet) à droite (volume maximum) dans les dispositions horizontale et verticale.
Navigation du contrôleur de lecture	Tab, Maj+Tab	Déplace le focus de bouton en bouton dans le composant contrôleur, où l'élément ayant le focus est mis en évidence. Cette navigation fonctionne avec les contrôles Pause/Lire, Rembobiner, Atteindre la fin, et les contrôles de volume désactivé et de volume maximum. Le focus se déplace de gauche à droite et de haut en bas lorsque les utilisateurs passent d'un élément à l'autre. L'utilisation de Maj+Tab déplace le focus de droite à gauche et de bas en haut. Lorsqu'il reçoit le focus via la touche Tab, le contrôle transmet immédiatement le focus au bouton Lire/Pause. Lorsque le focus est sur le bouton de volume maximum puis que la touche Tab est enfoncée, le focus se déplace vers le contrôle suivant dans l'ordre des tabulations sur la scène.
Bouton de contrôle donné	Espace ou Entrée/ Retour	Sélectionne l'élément ayant le focus. Lorsqu'il est enfoncé, le bouton apparaît dans son état enfoncé. Lorsqu'il est relâché, le bouton reprend son état survolé avec focus.

Présentation des composants média

Cette section fournit un aperçu du mode de fonctionnement des composants média. La plupart des propriétés répertoriées dans cette section peuvent être directement définies à l'aide de l'inspecteur des composants. (Voir la section « Utilisation de l'inspecteur des composants avec les composants média », à la page 874).

Les propriétés suivantes peuvent être définies pour les composants MediaDisplay et MediaPlayback, à l'exception des propriétés de disposition, qui sont présentées plus loin dans cette section :

- Le type de support, qui peut être défini sur MP3 ou FLV (voir `Media.mediaType` et `Media.setMedia()`).
- Le chemin relatif ou absolu du dossier, qui contient le fichier de support à lire en continu (voir `Media.contentPath`).
- Les objets point de repère, ainsi que leur nom, heure et propriétés de lecteur (voir `Media.addCuePoint()` et `Media.cuePoints`). Le nom du point de repère est aléatoire et doit être défini de manière à avoir un sens lors de l'utilisation d'événements d'écoute et de trace. Un point de repère diffuse un événement `cuePoint` lorsque la valeur de sa propriété time est égale à celle de l'emplacement de la tête de lecture du composant MediaPlayback ou MediaDisplay auquel il est associé. La propriété player est une référence à l'occurrence de MediaPlayback à laquelle elle est associée. Vous pouvez supprimer des points de repère en utilisant `Media.removeCuePoint()` et `Media.removeAllCuePoints()`.

Les composants média en flux continu diffusent plusieurs événements associés. Les événements de diffusion suivants peuvent être utilisés pour définir d'autres éléments en mouvement :

- Un événement `change` est diffusé en continu par les composants MediaDisplay et MediaPlayback pendant la lecture du support. (Voir `Media.change`.)
- Un événement `progress` est diffusé en continu par les composants MediaDisplay et MediaPlayback pendant le chargement du support. (Voir `Media.progress`.)
- Un événement `click` est diffusé par les composants MediaController et MediaPlayback chaque fois qu'un utilisateur clique sur le bouton Pause/Lire. Dans ce cas, la propriété `detail` de l'objet événement fournit des informations sur le bouton sur lequel l'utilisateur a cliqué. (Voir `Media.click`.)
- Un événement `volume` est diffusé par les composants MediaController et MediaPlayback lorsque l'utilisateur règle les contrôles de volume. (Voir `Media.volume`.)
- Un événement `playheadChange` est diffusé par les composants MediaController et MediaPlayback lorsque l'utilisateur déplace le curseur de lecture ou lorsqu'il clique sur les boutons Rembobiner ou Atteindre la fin. (Voir `Media.playheadChange`.)

Le composant MediaDisplay fonctionne en parallèle avec le composant MediaController. Lorsqu'ils sont combinés, leur comportement est identique à celui du composant MediaPlayback mais ils offrent davantage de flexibilité dans l'aspect de votre présentation.

Présentation du composant MediaDisplay

Lorsque vous placez un composant MediaDisplay sur la scène, il apparaît sans interface utilisateur. Il s'agit simplement d'un conteneur destiné à contenir et à lire des supports. Les propriétés suivantes affectent l'aspect des supports vidéos lus dans un composant MediaDisplay :

- `Media.aspectRatio`
- `Media.autoSize`
- Hauteur (dans l'inspecteur Propriétés)
- Largeur (dans l'inspecteur Propriétés)

REMARQUE	L'affichage est vide sauf si un support est en cours de lecture.

La propriété `Media.aspectRatio` est prioritaire par rapport aux autres propriétés. Lorsque la propriété `Media.aspectRatio` est définie sur `true` (valeur par défaut), le composant réajuste systématiquement la taille du support en cours de lecture pour conserver les proportions du support.

Pour les fichiers FLV, lorsque la propriété `Media.autoSize` est définie sur `true`, le support est affiché à sa taille préférée quelle que soit celle du composant. Ceci signifie que si la taille de l'occurrence de MediaDisplay est différente de celle du support, le support dépassera les limites de l'occurrence ou ne remplira pas la taille de l'occurrence. Lorsque la propriété `Media.autoSize` est définie sur `false`, Flash utilise la taille de l'occurrence autant que possible, tout en tenant compte des proportions. Si les propriétés `Media.autoSize` et `Media.aspectRatio` sont toutes deux définies sur `false`, la taille exacte du composant est utilisée.

REMARQUE	Etant donné qu'aucune image n'est affichée avec les fichiers MP3, il est inutile de définir `Media.autoSize`, car cela n'aura aucun effet. Pour les fichiers MP3, la taille minimum utilisable est de 60 pixels de hauteur sur 256 pixels de largeur dans l'orientation par défaut.

Le composant MediaDisplay prend également en charge la propriété `Media.volume`. Cette propriété prend des valeurs entières comprises entre 0 (muet) et 100 (volume maximum). Le paramètre par défaut est 75.

Présentation du composant MediaController

L'interface du composant MediaController dépend de ses propriétés `Media.controllerPolicy` et `Media.backgroundStyle`. La propriété `Media.controllerPolicy` détermine si le jeu de contrôles du support est toujours visible, réduit ou uniquement visible lorsque la souris passe sur la zone de contrôle du composant. Lorsqu'il est réduit, le contrôleur affiche une barre de progression modifiée, qui est une combinaison de la barre de chargement et de la barre de lecture. Elle affiche la progression des octets chargés en bas de la barre, et la progression de la tête de lecture est affichée juste au dessus. Lorsqu'il est développé, le contrôleur affiche une version étendue de la barre de lecture/chargement, qui contient les éléments suivants :

- Des étiquettes de texte sur la gauche indiquant l'état de la lecture (en flux continu ou en pause), et sur la droite l'emplacement de la tête de lecture en secondes
- Un indicateur d'emplacement de la tête de lecture
- Un curseur que les utilisateurs peuvent faire glisser pour naviguer dans le support

Le composant MediaController fournit également les éléments suivants :

- Un bouton Lire/Pause
- Des boutons Rembobiner et Atteindre la fin, qui naviguent respectivement vers le début et vers la fin du support
- Un contrôle de volume qui comprend un curseur, un bouton muet et un bouton volume maximum

Les états réduit et développé du composant MediaController utilisent tous deux la propriété `Media.backgroundStyle`. Cette propriété indique si le contrôleur dessine un arrière-plan chromé (par défaut) ou s'il autorise l'affichage de l'arrière-plan du support depuis les contrôles.

Le composant MediaController possède un paramètre d'orientation, `Media.horizontal`, que vous pouvez utiliser pour dessiner le composant avec une orientation horizontale (par défaut) ou verticale. Dans une orientation horizontale, la barre de lecture suit le support en cours de lecture de la gauche vers la droite. Dans une orientation verticale, la barre de lecture suit le support de bas en haut.

Vous pouvez combiner les composants MediaDisplay et MediaController à l'aide des méthodes `Media.associateDisplay()` et `Media.associateController()`. Ces méthodes permettent à l'occurrence de MediaController de mettre à jour ses contrôles en fonction des événements diffusés depuis l'occurrence de MediaDisplay. Elles permettent également au composant MediaDisplay de réagir aux paramètres utilisateur dans le composant MediaController.

Présentation du composant MediaPlayback

Le composant MediaPlayback contient les sous-composants MediaController et MediaDisplay. Les portions de MediaController et de MediaDisplay sont toujours redimensionnées pour correspondre à la taille de l'occurrence générale du composant MediaPlayback.

Le composant MediaPlayback utilise `Media.controlPlacement` pour définir la disposition des contrôles. En définissant cette propriété sur `top`, `bottom`, `left` ou `right`, vous pouvez indiquer l'endroit où les contrôles sont tracés par rapport à l'affichage. Par exemple, la valeur `right` donne une orientation verticale au contrôle et le positionne sur la droite de l'affichage.

Utilisation des composants média

L'utilisation croissante de supports pour fournir des informations aux utilisateurs Web implique la volonté de nombreux développeurs de fournir à ces mêmes utilisateurs un moyen de lire ces supports en flux continu et de les contrôler. Vous pouvez utiliser des composants média dans les types de situation suivants :

- Affichage d'un support présentant une société
- Lecture en flux continu d'animations ou d'aperçus d'animations
- Lecture en continu de chansons ou de fragments de chansons
- Fourniture de matériel de formation par le biais de supports

Utilisation du composant MediaPlayback

Admettons que vous devez développer un site Web permettant aux utilisateurs de prévisualiser des DVD et des CD que vous vendez dans un environnement de supports enrichis. L'exemple suivant indique les étapes du processus. (Il suppose que vous pouvez insérer des composants en flux continu dans votre site Web.)

Pour créer un document Flash qui affiche un aperçu de CD ou de DVD :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Ouvrez le panneau Composants et double-cliquez sur le composant MediaPlayback pour en placer une occurrence sur la scène.
3. Sélectionnez l'occurrence du composant MediaPlayback et entrez le nom d'occurrence **myMedia** dans l'inspecteur Propriétés.
4. Dans l'inspecteur de composants, définissez le type de votre support en fonction du type de support qui sera diffusé en flux continu (MP3 ou FLV).

5. Si vous avez sélectionné FLV, entrez la durée de la vidéo dans les zones de texte Video Length (Durée de la vidéo). Utilisez le format *HH:MM:SS*.
6. Entrez l'emplacement de votre aperçu vidéo dans la zone de texte URL. Par exemple, vous pouvez entrer **http://www.helpexamples.com/flash/video/clouds.flv**.
7. Définissez les options souhaitées pour les cases à cocher Automatically Play (Lecture automatique), Use Preferred Media Size (Utiliser la taille de support préférée) et Respect Aspect Ratio (Respecter les proportions).
8. Définissez l'emplacement du contrôle du côté désiré du composant MediaPlayback.
9. En cliquant sur le bouton Ajouter +, ajoutez un point de repère vers la fin du support. Ce point de repère est utilisé avec un écouteur pour ouvrir une fenêtre contextuelle informant l'utilisateur que l'animation est en vente. Nommez le point de repère **cuePointName** et placez-le vers la fin de votre support.
10. Faites glisser un composant Window du panneau Composants jusqu'à la bibliothèque du document actuel.

    Cette procédure place un symbole appelé Window dans votre bibliothèque, puis met le composant Window à la disposition de votre SWF lors de l'exécution.
11. Créez une zone de texte et entrez un texte informant l'utilisateur que l'animation est à vendre.
12. Choisissez Modification > Convertir en symbole pour convertir la zone de texte en clip, puis nommez-le **mySale_mc**.
13. Cliquez avec le bouton droit de la souris (Windows) ou appuyez sur la touche Contrôle (Macintosh) sur le clip `mySale_mc` dans la bibliothèque, sélectionnez Liaison, puis Exporter pour ActionScript. Assurez-vous que l'identificateur de liaison est correctement nommé.

    Un clip est placé dans la bibliothèque d'exécution.
14. Supprimez la zone de texte de la scène.

    La zone de texte est nécessaire uniquement dans la bibliothèque.
15. Ajoutez le code ActionScript suivant à l'image 1. Ce code crée un écouteur qui ouvre une fenêtre contextuelle informant l'utilisateur que l'animation est en vente.

```
// Importation des classes nécessaires pour créer dynamiquement
// la fenêtre contextuelle.

import mx.containers.Window;
import mx.managers.PopUpManager;

// Création d'un objet écouteur pour ouvrir la fenêtre contextuelle
// de vente.
var saleListener:Object = new Object();
```

```
saleListener.cuePoint = function(eventObj:Object) {

var saleWin:MovieClip = PopUpManager.createPopUp(_root, Window, false,
  {closeButton:true, title:"Movie Sale", contentPath:"mySale_mc"});

// Agrandissement de la fenêtre pour que le contenu puisse être inclus.

saleWin.setSize(80, 80);
var delSaleWin:Object = new Object();
delSaleWin.click = function(eventObj:Object) {
  saleWin.deletePopUp();
}
saleWin.addEventListener("click", delSaleWin);

}

myMedia.addEventListener("cuePoint", saleListener);
```

16. Sélectionnez Contrôle > Tester l'animation pour tester le fichier SWF.

    Lorsque l'application atteint la lecture du point de repère cuePointName, une fenêtre s'ouvre pour afficher votre message.

Utilisation des composants MediaDisplay et MediaController

Vous pouvez utiliser conjointement les composants MediaDisplay et MediaController pour avoir davantage de contrôle sur l'aspect de l'affichage de votre support. L'exemple suivant crée une application Flash qui affiche le support d'aperçu de votre CD et DVD.

Pour créer un document Flash qui affiche un aperçu de CD ou de DVD :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. A partir du panneau Composants, faites glisser un composant MediaController *et* un composant MediaDisplay sur la scène.
3. Sélectionnez l'occurrence MediaDisplay et nommez-la **myDisplay** dans l'inspecteur Propriétés.
4. Sélectionnez l'occurrence MediaController et nommez-la **myController** dans l'inspecteur Propriétés.
5. Sélectionnez l'occurrence MediaDisplay, ouvrez l'inspecteur de composants, puis cliquez sur l'onglet Paramètres. Définissez le type de votre support en fonction du type de support qui sera diffusé en flux continu (MP3 ou FLV).
6. Si vous avez sélectionné FLV, entrez la durée de la vidéo dans les zones de texte Video Length (Durée de la vidéo). Utilisez le format *HH:MM:SS*.

7. Entrez l'emplacement de votre aperçu vidéo dans la zone de texte URL. Par exemple, vous pouvez entrer **www.helpexamples.com/flash/video/clouds.flv**.
8. Définissez les options souhaitées pour les cases à cocher Automatically Play (Lecture automatique), Use Preferred Media Size (Utiliser la taille de support préférée) et Respect Aspect Ratio (Respecter les proportions).
9. Sélectionnez l'occurrence MediaController, puis, dans l'inspecteur de composants, cliquez sur l'onglet Paramètres et choisissez une orientation verticale en définissant la propriété `horizontal` sur `false`.
10. Dans l'inspecteur de composants, cliquez sur l'onglet Paramètres et définissez `backgroundStyle` sur `none`.

    Ceci spécifie que l'occurrence de MediaController ne doit pas dessiner d'arrière-plan mais remplir le support entre les contrôles.

    Utilisez ensuite un comportement pour associer les occurrences de MediaController et de MediaDisplay de sorte que l'occurrence de MediaController reflète précisément les déplacements de la tête de lecture ainsi que d'autres paramètres de l'occurrence de MediaDisplay, et que l'occurrence de MediaDisplay réponde aux clics de l'utilisateur.
11. L'occurrence MediaController étant toujours sélectionnée, ouvrez le panneau Comportements (Fenêtre > Comportements).
12. Dans le panneau Comportements, cliquez sur Ajouter (+), puis choisissez Media (Médias) > Affichage associé.
13. Dans la fenêtre Affichage associé, sélectionnez `myDisplay` sous `_root`, puis cliquez sur OK.

Pour plus d'informations sur l'utilisation des comportements avec les composants média, reportez-vous à « Contrôle des composants média à l'aide des comportements », à la page 876.

Utilisation de l'inspecteur des composants avec les composants média

L'inspecteur des composants facilite notamment la définition des paramètres et des propriétés des composants média. Pour utiliser ce panneau, cliquez sur le composant souhaité sur la scène et, l'inspecteur Propriétés étant ouvert, cliquez sur Ouvrir l'inspecteur des composants. L'inspecteur des composants peut être utilisé de différentes manières :

- Pour lire automatiquement le support (reportez-vous à `Media.activePlayControl` et `Media.autoPlay`)
- Pour conserver ou ignorer les proportions du support (voir `Media.aspectRatio`)
- Pour déterminer si le support est automatiquement redimensionné en fonction de la taille de l'occurrence de composant (voir `Media.autoSize`)

- Pour activer ou désactiver l'arrière-plan chromé (voir `Media.backgroundStyle`)
- Pour spécifier le chemin de votre support sous la forme d'une URL (voir `Media.contentPath`)
- Pour spécifier la visibilité des contrôles de lecture (voir `Media.controllerPolicy`)
- Pour ajouter des objets point de repère (voir `Media.addCuePoint()`)
- Pour supprimer des objets point de repère (voir `Media.removeCuePoint()`)
- Pour définir l'orientation des occurrences MediaController (voir `Media.horizontal`)
- Pour définir le type de support lu (voir `Media.setMedia()`)
- Pour définir la durée de lecture du support FLV (voir `Media.totalTime`)
- Pour définir les derniers chiffres du temps affiché en indiquant des millisecondes ou des images par seconde (ips)

Vous devez vous familiariser avec certains concepts lorsque vous utilisez l'inspecteur des composants :

- Le contrôle de durée de la vidéo n'est pas disponible lorsque vous sélectionnez le type de vidéo MP3 car les informations sont lues automatiquement lorsque des fichiers MP3 sont utilisés. Pour les fichiers FLV créés avec Flash Video Exporter 1.0, vous devez entrer le temps total du support (`Media.totalTime`) pour que la barre de lecture du composant MediaPlayback (ou de tout composant MediaController d'écoute) puisse refléter précisément la progression de la lecture. Les fichiers FLV créés avec Flash Video Exporter 1.1 ou une version ultérieure définissent la durée automatiquement.
- Si vous définissez le type de fichier sur FLV, une option Milliseconds s'affiche et (si l'option Milliseconds est désactivée) un menu contextuel FPS apparaît. Lorsque cette option est activée, le contrôle FPS est masqué. Dans ce mode, la durée affichée dans la barre de lecture lors de l'exécution est au format *HH:MM:SS.mmm* (*H* = heures, *M* = minutes, *S* = secondes, *m* = millisecondes) et les points de repère sont définis en secondes. Lorsque l'option est désactivée, le contrôle FPS est activé et la durée affichée dans la barre de lecture est au format *HH:MM:SS.II* (*I* = images par seconde), alors que les points de repère sont définis par image.

REMARQUE	Vous pouvez uniquement définir la valeur FPS en utilisant l'inspecteur des composants. La définition d'une valeur fps à l'aide d'ActionScript n'a aucun effet et est ignorée.

Contrôle des composants média à l'aide des comportements

Les comportements sont des scripts ActionScript prêts à l'emploi ajoutés à une occurrence, tel qu'un composant MediaDisplay, pour contrôler cet objet. Les comportements vous permettent d'ajouter la puissance, le contrôle et la flexibilité du codage ActionScript à votre document sans avoir à créer le code ActionScript vous-même.

Pour contrôler un composant média à l'aide d'un comportement, vous devez utiliser le panneau Comportements afin d'appliquer le comportement à une occurrence de composant média donnée. Définissez l'événement déclencheur du comportement (par exemple le fait d'atteindre un point de repère spécifique), sélectionnez un objet cible (les composants média qui sont affectés par le comportement) et, si nécessaire, sélectionnez les paramètres du comportement (par exemple le clip dans le support vers lequel naviguer).

Le tableau ci-dessous présente les comportements disponibles dans Flash qui permettent de contrôler les composants média intégrés.

Comportement	Objectif	Paramètres
Contrôleur associé	Associe un composant MediaController à un composant MediaDisplay	Nom d'occurrence des composants MediaController cibles
Affichage associé	Associe un composant MediaDisplay à un composant MediaController	Nom d'occurrence des composants MediaController cibles
Labeled Frame CuePoint Navigation (Exploration des points de repère de l'image étiquetée)	Place une action sur une occurrence de MediaDisplay ou de MediaPlayback qui indique à un clip spécifié de naviguer vers une image portant le même nom qu'un point de repère donné	Nom de l'image et nom du point de repère (les noms doivent être identiques)
Slide CuePoint Navigation (Exploration des points de repère de la diapositive)	Fait naviguer un document Flash basé sur des diapositives jusqu'à une diapositive portant le même nom qu'un point de repère donné	Nom de la diapositive et nom du point de repère (les noms doivent être identiques)

Pour associer un composant MediaDisplay à un composant MediaController :

1. Assurez-vous que vos paramètres de publication sont bien définis sur ActionScript 2.0.
2. Placez une occurrence MediaDisplay et une occurrence MediaController sur la scène.
3. Sélectionnez l'occurrence MediaDisplay et, dans l'inspecteur Propriétés, nommez-la **myMediaDisplay**.
4. Sélectionnez l'occurrence MediaController qui déclenche le comportement.
5. Dans le panneau Comportements, cliquez sur Ajouter (+) et sélectionnez Media (Médias) > Affichage associé.
6. Dans la fenêtre Affichage associé, sélectionnez `myMediaDisplay` sous `_root`, puis cliquez sur OK.

REMARQUE	Si vous avez associé le composant MediaDisplay au composant MediaController, il n'est pas nécessaire d'associer le composant MediaController au composant MediaDisplay.

Pour associer un composant MediaController à un composant MediaDisplay :

1. Assurez-vous que vos paramètres de publication sont bien définis sur ActionScript 2.0.
2. Placez une occurrence MediaDisplay et une occurrence MediaController sur la scène.
3. Sélectionnez l'occurrence MediaController et, dans l'inspecteur Propriétés, nommez-la **myMediaController**.
4. Sélectionnez l'occurrence de MediaDisplay qui déclenche le comportement.
5. Dans le panneau Comportements, cliquez sur Ajouter (+) et sélectionnez Media (Médias) > Contrôleur associé.
6. Dans la fenêtre Contrôleur associé, sélectionnez `myMediaController` sous `_root`, puis cliquez sur OK.

Pour utiliser un comportement Labeled Frame CuePoint Navigation (Exploration des points de repère de l'image étiquetée) :

1. Assurez-vous que vos paramètres de publication sont bien définis sur ActionScript 2.0.
2. Placez une occurrence du composant MediaDisplay ou MediaPlayback sur la scène.
3. Sélectionnez l'image vers laquelle vous souhaitez que le support navigue, puis dans l'inspecteur Propriétés, entrez le nom d'image **myLabeledFrame**.
4. Sélectionnez votre occurrence de MediaDisplay ou de MediaPlayback.

5. Dans l'inspecteur de composants, cliquez sur le bouton Ajouter (+), entrez l'heure du point de repère au format *HH:MM:SS:mmm* ou *HH:MM:SS:II*, puis nommez le point de repère **myLabeledFrame**.

   Le point de repère indique la durée devant s'écouler avant que vous ne naviguiez vers l'image sélectionnée. Par exemple, si vous souhaitez atteindre `myLabeledFrame` 5 secondes dans le support, entrez 5 dans la zone de texte SS, puis **myLabeledFrame** dans la zone de texte Nom.
6. Dans le panneau Comportements, cliquez sur Ajouter (+) et sélectionnez Media (Médias) > Labeled Frame CuePoint Navigation (Exploration des points de repère de l'image étiquetée).
7. Dans la fenêtre Labeled Frame CuePoint Navigation (Exploration des points de repère de l'image étiquetée), sélectionnez le clip `_root`, puis cliquez sur OK.

Pour utiliser un comportement Slide CuePoint Navigation (Exploration des points de repère d'une diapositive) :

1. Ouvrez un diaporama Flash existant. (Flash CS3 ne prend pas en charge la création de fichiers FLA de présentation.)
2. Placez une occurrence du composant MediaDisplay ou MediaPlayback sur la scène.
3. Dans le panneau situé à gauche de la scène, cliquez sur le bouton Insérer un écran (+) pour ajouter une deuxième diapositive, puis sélectionnez la deuxième diapositive et renommez-la **mySlide**.
4. Sélectionnez votre occurrence de MediaDisplay ou de MediaController.
5. Dans l'inspecteur de composants, cliquez sur le bouton Ajouter (+), entrez l'heure du point de repère au format *HH:MM:SS:mmm* ou *HH:MM:SS:II*, puis nommez le point de repère **MySlide**.

   Le point de repère indique la durée devant s'écouler avant que vous ne naviguiez vers la diapositive sélectionnée. Par exemple, si vous souhaitez atteindre `mySlide` 5 secondes dans le support, entrez 5 dans la zone de texte SS, puis **mySlide** dans la zone de texte Nom.
6. Dans le panneau Comportements, cliquez sur Ajouter (+) et sélectionnez Media (Médias) > Slide CuePoint Navigation (Exploration des points de repère d'une diapositive).
7. Dans la fenêtre Slide CuePoint Navigation (Exploration des points de repère d'une diapositive), sélectionnez `Présentation` sous le clip `_root`, puis cliquez sur OK.

Paramètres des composants média

Les tableaux suivants répertorient les paramètres de création MediaDisplay, MediaController et MediaPlayback que vous pouvez définir pour une occurrence du composant média donnée dans l'inspecteur Propriétés.

Paramètres de MediaDisplay

Nom	Type	Valeur par défaut	Description
Automatically Play (Lecture automatique) (`Media.autoPlay`)	Valeur booléenne	Sélectionné (Selected)	Indique si le support est lu dès la fin de son chargement.
Use Preferred Media Size (Utiliser la taille de support préférée) (`Media.autoSize`)	Valeur booléenne	Sélectionné (Selected)	Indique si le support associé à l'occurrence de MediaDisplay correspond à la taille du composant ou utilise simplement sa taille par défaut.
FPS	Nombre entier	30	Indique le nombre d'images par seconde. Lorsque l'option Milliseconds (Millisecondes) est activée, ce contrôle est désactivé.
Cue Points (Points de repère) (`Media.cuePoints`)	Tableau	Non défini (Undefined)	Tableau d'objets point de repère, chaque objet possédant un nom et une position dans le temps dans un format valide *HH:MM:SS:ll* (option Milliseconds [Millisecondes] activée) ou *HH:MM:SS:mmm*.
FLV ou MP3 (`Media.mediaType`)	FLV ou MP3	`FLV`	Indique le type de support à lire.
Milliseconds (Millisecondes)	Valeur booléenne	Unselected	Indique si la barre de lecture utilise des images ou des millisecondes et si les points de repère utilisent des secondes ou des images. Lorsque cette option est activée, le contrôle FPS est masqué.

Nom	Type	Valeur par défaut	Description
URL (`Media.contentPath`)	Chaîne	Non défini (Undefined)	Chaîne contenant le chemin et le nom de fichier du support à lire.
Video Length (Durée de la vidéo) (`Media.totalTime`)	Nombre entier	Non défini (Undefined)	Le temps total nécessaire pour lire le support FLV. Ce paramètre est requis pour que la barre de lecture fonctionne correctement. Ce contrôle est uniquement visible lorsque le type de support est défini sur FLV.

Paramètres de MediaController

Nom	Type	Valeur par défaut	Description
activePlayControl (`Media.activePlayControl`)	Chaîne : `pause` ou `play`	`pause`	Détermine si la barre de lecture est en mode pause ou play au moment de l'instanciation. Ce mode détermine l'image affichée sur le bouton Lire/Pause qui est l'opposé de l'état de lecture/pause réel du contrôleur.
backgroundStyle (`Media.backgroundStyle`)	Chaîne : `default` ou `none`	`default`	Indique si l'arrière-plan chromé est dessiné pour l'occurrence de MediaController.
controllerPolicy (`Media.controllerPolicy`)	Chaîne : `auto`, `on` ou `off`	`auto`	Détermine si le contrôleur s'ouvre ou se ferme en fonction de la position de la souris, ou est verrouillé dans l'état ouvert ou fermé.
horizontal (`Media.horizontal`)	Valeur booléenne	`true`	Détermine si la portion de contrôleur de l'occurrence est orientée verticalement ou horizontalement. Une valeur `true` indique que le composant a une position horizontale.

Nom	Type	Valeur par défaut	Description
enabled (activé)	Valeur booléenne	`true`	Détermine si ce contrôle peut être modifié par l'utilisateur. Une valeur `true` indique que le contrôle peut être modifié.
visible	Valeur booléenne	`true`	Détermine si ce contrôle peut être affiché par l'utilisateur. Une valeur `true` indique que le contrôle peut être affiché.

Paramètres de MediaPlayback

Nom	Type	Valeur par défaut	Description
Control Placement (Emplacement de contrôle) (`Media.controlPlacement`)	Chaîne : `top`, `bottom`, `left`, `right`	`bottom`	Position du contrôleur. La valeur est liée à l'orientation.
Control Visibility (Visibilité de contrôle) (`Media.controllerPolicy`)	Valeur booléenne	true	Détermine si le contrôleur s'ouvre ou se ferme en fonction de la position de la souris.
Automatically Play (Lecture automatique) (`Media.autoPlay`)	Valeur booléenne	true	Indique si le support est lu dès la fin de son chargement.
Use Preferred Media Size (Utiliser la taille de support préférée) (`Media.autoSize`)	Valeur booléenne	true	Détermine si l'occurrence de MediaController est redimensionnée en fonction de la taille du support ou si elle utilise d'autres paramètres.
FPS	Nombre entier	30	Nombre d'images par seconde. Lorsque l'option Milliseconds (Millisecondes) est activée, ce contrôle est désactivé.
Cue Points (Points de repère) (`Media.cuePoints`)	Tableau	Non défini (Undefined)	Tableau d'objets point de repère, chaque objet possédant un nom et une position dans le temps dans un format valide *HH:MM:SS:mmm* (option Milliseconds [Millisecondes] activée) ou *HH:MM:SS:II*.

Nom	Type	Valeur par défaut	Description
FLV ou MP3 (`Media.mediaType`)	Chaîne : FLV ou MP3	`FLV`	Indique le type de support à lire.
Milliseconds (Millisecondes)	Valeur booléenne	false	Indique si la barre de lecture utilise des images ou des millisecondes et si les points de repère utilisent des secondes ou des images. Lorsque cette option est activée, le contrôle FPS est désactivé.
URL (`Media.contentPath`)	Chaîne	Non défini (Undefined)	Chaîne contenant le chemin et le nom de fichier du support à lire.
Video Length (Durée de la vidéo) (`Media.totalTime`)	Nombre entier	Non défini (Undefined)	Le temps total nécessaire pour lire le support FLV. Ce paramètre est requis pour que la barre de lecture fonctionne correctement.

Création d'applications à l'aide des composants média

Il est relativement facile de créer du contenu Flash à l'aide des composants média et ce, en quelques étapes seulement. Cet exemple montre comment créer une application pour lire un petit fichier de support accessible au public.

Pour ajouter un composant média à une application :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Dans le panneau Composants, double-cliquez sur le composant MediaPlayback pour l'ajouter à la scène.
3. Dans l'inspecteur Propriétés, procédez comme suit :
   - Entrez le nom d'occurrence **myMedia**.
   - Cliquez sur Ouvrir l'inspecteur des composants.
4. Dans l'inspecteur de composants, entrez**http://www.helpexamples.com/flash/video/water.flv** dans la zone de texte URL.
5. Choisissez Contrôle > Tester l'animation pour visualiser la lecture du support.

Personnalisation des composants média

Vous pouvez appliquer des enveloppes pour modifier l'apparence de vos composant médias. Pour approfondir votre connaissance sur la personnalisation des composants, reportez-vous à Chapitre 5, « Personnalisation des composants » dans *Utilisation des composants ActionScript 2.0*.

Utilisation des styles avec les composants média

Les composants média n'utilisent pas les styles.

Utilisation d'enveloppes avec les composants média

Les composants média ne prennent pas en charge le réhabillage dynamique, même si vous pouvez ouvrir le document source des composants média et modifier leurs actifs pour obtenir l'aspect escompté. Il est préférable de faire une copie de ce fichier et d'utiliser cette dernière, de façon à pouvoir toujours revenir au document source installé. Le document source des composants média se trouve aux emplacements suivants :

- Windows : C:\Program Files\Adobe\Adobe Flash CS3*langue*\Configuration\Component Source\ActionScript 2.0\FLA\MediaComponents.fla
- Macintosh : HD/Applications/Adobe/Adobe Flash CS3/Configuration/Component Source/ActionScript 2.0/FLA/MediaComponents.fla

Classe Media

Héritage MovieClip > Classe UIObject > Classe UIComponent > Media

Noms de classe ActionScript mx.controls.MediaController, mx.controls.MediaDisplay, mx.controls.MediaPlayback

Toutes les classes de composants ont une propriété `version` qui correspond à une propriété de classe. Les propriétés de classe sont uniquement disponibles pour la classe elle-même.
La propriété `version` renvoie une chaîne qui indique la version du composant. Pour accéder à cette propriété, utilisez le code suivant :

```
trace(mx.controls.MediaPlayback.version);
```

REMARQUE

Le code `trace(myMediaInstance.version);` renvoie `undefined`.

Méthodes de la classe Media

Le tableau suivant répertorie les méthodes de la classe Media.

Méthode	Composants	Description
`Media.addCuePoint()`	MediaDisplay, MediaPlayback	Ajoute un objet point de repère à l'occurrence du composant.
`Media.associateController()`	MediaDisplay	Associe une occurrence de MediaDisplay à une occurrence de MediaController.
`Media.associateDisplay()`	MediaController	Associe une occurrence de MediaController à une occurrence de MediaDisplay.
`Media.displayFull()`	MediaPlayback	Convertit l'occurrence du composant en mode de lecture plein écran.
`Media.displayNormal()`	MediaPlayback	Reconvertit l'occurrence de composant à sa taille d'écran originale.
`Media.getCuePoint()`	MediaDisplay, MediaPlayback	Renvoie un objet point de repère.
`Media.pause()`	MediaDisplay, MediaPlayback	Met la tête de lecture en pause à son emplacement actuel dans le scénario du support.
`Media.play()`	MediaDisplay, MediaPlayback	Lit le support associé à l'occurrence de composant à un point de départ donné.
`Media.removeAllCuePoints()`	MediaDisplay, MediaPlayback	Supprime tous les objets point de repère associés à une occurrence de composant donnée.
`Media.removeCuePoint()`	MediaDisplay, MediaPlayback	Supprime un point de repère spécifié associé à une occurrence de composant donnée.
`Media.setMedia()`	MediaDisplay, MediaPlayback	Définit le type et le chemin du support sur le type de support spécifié.
`Media.stop()`	MediaDisplay, MediaPlayback	Arrête la tête de lecture et la déplace à la position 0, qui correspond au début du support.

Propriétés de la classe Media

Le tableau suivant répertorie les propriétés de la classe Media.

Propriété	Composants	Description
`Media.activePlayControl`	MediaController	Détermine l'état du composant lors de son chargement à l'exécution.
`Media.aspectRatio`	MediaDisplay, MediaPlayback	Détermine si l'occurrence du composant conserve les proportions de sa vidéo.
`Media.autoPlay`	MediaDisplay, MediaPlayback	Détermine si l'occurrence du composant est immédiatement mise en mémoire tampon et lue.
`Media.autoSize`	MediaDisplay, MediaPlayback	Détermine la taille de la portion d'affichage de support du composant MediaDisplay ou MediaPlayback.
`Media.backgroundStyle`	MediaController	Détermine si l'occurrence du composant dessine son arrière-plan chromé.
`Media.bytesLoaded`	MediaDisplay, MediaPlayback	Lecture seule ; le nombre d'octets chargés pouvant être lus.
`Media.bytesTotal`	MediaDisplay, MediaPlayback	Nombre d'octets à charger dans l'occurrence du composant.
`Media.contentPath`	MediaDisplay, MediaPlayback	Chaîne contenant le chemin relatif et le nom de fichier du support devant être diffusé en flux continu et lu.
`Media.controllerPolicy`	MediaController, MediaPlayback	Détermine si le contrôleur est masqué lorsqu'il est instancié et n'est affiché que lorsque l'utilisateur fait passer la souris sur l'état réduit du contrôleur.
`Media.controlPlacement`	MediaPlayback	Détermine l'endroit où les contrôles du composants sont positionnés.
`Media.cuePoints`	MediaDisplay, MediaPlayback	Tableau d'objets point de repère ayant été associés à une occurrence de composant donnée.
`Media.horizontal`	MediaController	Détermine l'orientation de l'occurrence du composant.
`Media.mediaType`	MediaDisplay, MediaPlayback	Détermine le type de support à lire.
`Media.playheadTime`	MediaDisplay, MediaPlayback	Contient la position actuelle de la tête de lecture (en secondes) pour le scénario de support en cours de lecture.

Propriété	Composants	Description
Media.playing	MediaDisplay, MediaPlayback, MediaController	Pour MediaDisplay et MediaPlayback, cette propriété est en lecture seule et renvoie une valeur booléenne pour indiquer si une occurrence de composant donnée lit un support. Pour MediaController, cette propriété a un accès en lecture/écriture et contrôle l'image (en cours de lecture ou en pause) affichée sur le bouton Lire/Pause du contrôleur.
Media.preferredHeight	MediaDisplay, MediaPlayback	Valeur par défaut de la hauteur d'un fichier FLV.
Media.preferredWidth	MediaDisplay, MediaPlayback	Valeur par défaut de la largeur d'un fichier FLV.
Media.totalTime	MediaDisplay, MediaPlayback	Nombre entier indiquant la durée totale du support, en secondes.
Media.volume	MediaDisplay, MediaPlayback	Nombre entier de 0 (minimum) à 100 (maximum) représentant le niveau de volume.

Evénements de la classe Media

Le tableau suivant répertorie les événements de la classe Media.

Evénement	Composants	Description
Media.change	MediaDisplay, MediaPlayback	Diffusé en continu pendant la lecture du support.
Media.click	MediaController, MediaPlayback	Diffusé lorsque l'utilisateur clique sur le bouton Lire/Pause.
Media.complete	MediaDisplay, MediaPlayback	Notification indiquant que la tête de lecture a atteint la fin du support.
Media.cuePoint	MediaDisplay, MediaPlayback	Notification indiquant que la tête de lecture a atteint un point de repère donné.
Media.playheadChange	MediaController, MediaPlayback	Diffusé par l'occurrence du composant lorsque l'utilisateur déplace le curseur de lecture ou clique sur le bouton Rembobiner ou Atteindre la fin.
Media.progress	MediaDisplay, MediaPlayback	Généré en continu jusqu'à la fin du téléchargement du support.

Evénement	Composants	Description
`Media.scrubbing`	MediaController, MediaPlayback	Généré lorsque l'utilisateur déplace la tête de lecture.
`Media.volume`	MediaController, MediaPlayback	Diffusé lorsque l'utilisateur règle le volume.

Media.activePlayControl

S'applique à

MediaController

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

myMedia`.activePlayControl`

Description

Propriété : une valeur de chaîne spécifiant l'état dans lequel le composant MediaController devrait être lorsqu'il est chargé à l'exécution. Une valeur « `play` » indique un état de lecture ; une valeur « `pause` » indique un état en pause. Définissez cette propriété et la propriété `autoPlay` pour qu'elles indiquent le même état. La valeur par défaut est « `play` ».

L'image du bouton affichée dans le composant MediaController est l'opposée de l'état de lecture/pause en cours. Par exemple, dans l'état de lecture, le MediaController affiche un bouton de pause qui résulterait du clic de l'utilisateur sur le bouton et du changement d'état.

Etant donné qu'elle indique l'état du contrôleur à la fin du chargement, vous devez définir la propriété `activePlayControl` avant de créer le contrôleur. Pour ce faire, utilisez l'inspecteur Propriétés ou l'inspecteur de composants si le composant est sur la scène. Si vous créez le composant avec le code ActionScript, vous devez définir cette propriété dans le paramètre `initObj`. La modification de la valeur de cette propriété une fois que le composant a été créé n'a aucune incidence. L'utilisateur peut changer la valeur uniquement en cliquant sur le bouton Lire/Pause.

Exemple

L'exemple suivant indique que le contrôle est en pause lors de son premier chargement :

```
myMedia.activePlayControl = "pause";
```

Voir aussi

`Media.autoPlay`

Media.addCuePoint()

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
myMedia.addCuePoint(cuePointName, cuePointTime)
```

Paramètres

`cuePointName` Chaîne qui nomme le point de repère.

`cuePointTime` Nombre, en secondes, indiquant le moment de diffusion d'un événement `cuePoint`.

Valeur renvoyée

Aucune.

Description

Méthode : ajoute un objet point de repère à une occurrence MediaPlayback ou MediaDisplay. Lorsque l'heure de la tête de lecture est égale à celle d'un point de repère, un événement `cuePoint` est diffusé.

Exemple

Le code suivant ajoute un point de repère nommé `Homerun` à `myMedia` à l'heure 16 secondes.

```
myMedia.addCuePoint("Homerun", 16);
```

Voir aussi

`Media.cuePoint`, `Media.cuePoints`, `Media.getCuePoint()`, `Media.removeAllCuePoints()`, `Media.removeCuePoint()`

Media.aspectRatio

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
myMedia.aspectRatio
```

Description

Propriété : une valeur booléenne indiquant si une occurrence de MediaDisplay ou de MediaPlayback conserve les proportions de sa vidéo pendant la lecture. Une valeur `true` indique que les proportions doivent être conservées, et une valeur `false` indique que les proportions peuvent changer pendant la lecture. La valeur par défaut est `true`.

Exemple

L'exemple suivant indique que les proportions peuvent changer lors de la lecture :

```
myMedia.aspectRatio = false;
```

Media.associateController()

S'applique à

MediaDisplay

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
myMedia.associateController(instanceName)
```

Paramètres

instanceName Chaîne indiquant le nom d'occurrence du composant MediaController à associer.

Valeur renvoyée

Aucune.

Description

Méthode : associe une occurrence de MediaDisplay à une occurrence de MediaController.

Si vous utilisez `Media.associateDisplay()` pour associer une occurrence MediaController à une occurrence MediaDisplay, vous n'êtes pas tenu d'utiliser `Media.associateController()`.

Exemple

Le code suivant associe `myMedia` à `myController` :

```
myMedia.associateController(myController);
```

Voir aussi

`Media.associateDisplay()`

Media.associateDisplay()

S'applique à

MediaController

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
myMedia.associateDisplay(instanceName)
```

Paramètres

instanceName Chaîne indiquant le nom d'occurrence du composant MediaDisplay à associer.

Valeur renvoyée

Aucune.

Description

Méthode : associe une occurrence de MediaController à une occurrence de MediaDisplay.

Si vous associez une occurrence MediaDisplay à une occurrence MediaController à l'aide de `Media.associateController()`, vous n'êtes pas tenu d'utiliser `Media.associateDisplay()`.

Exemple

Le code suivant associe `myMedia` à `myDisplay` :

```
myMedia.associateDisplay(myDisplay);
```

Voir aussi

`Media.associateController()`

Media.autoPlay

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`myMedia``.autoPlay`

Description

Propriété : une valeur booléenne indiquant si une occurrence de MediaPlayback ou de MediaDisplay doit immédiatement être mise en mémoire tampon et lue. Une valeur `true` indique que le contrôle est mis en mémoire tampon et lu à l'exécution alors qu'une valeur `false` indique que le contrôle est arrêté à l'exécution. Cette propriété dépend des propriétés `contentPath` et `mediaType`. Si les propriétés `contentPath` et `mediaType` ne sont pas définies, il n'y a pas de lecture lors de l'exécution. La valeur par défaut est `true`.

Exemple

L'exemple suivant indique que le contrôle est en pause lors de son premier chargement :

```
myMedia.autoPlay = false;
```

Voir aussi

`Media.contentPath`, `Media.mediaType`

Media.autoSize

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

myMedia.autoSize

Description

Propriété : une valeur booléenne indiquant la taille de la zone d'affichage du support du composant MediaDisplay ou MediaPlayback.

Pour le composant MediaDisplay, la propriété se comporte de la manière suivante :

- Si vous définissez cette propriété sur `true`, Flash affiche le support à sa taille préférée quelle que soit la taille du composant. Ceci implique que, à moins que la taille de l'occurrence MediaDisplay ne soit la même que celle du support, le support dépasse les limites de l'occurrence ou ne remplit pas l'occurrence.
- Si vous définissez cette propriété sur `false`, Flash utilise la taille de l'occurrence dans la mesure du possible, tout en respectant les proportions. Si les propriétés `Media.autoSize` et `Media.aspectRatio` sont toutes deux définies sur `false`, la taille exacte du composant est utilisée.

Pour le composant MediaPlayback, la propriété se comporte de la manière suivante :

- Si vous définissez cette propriété sur `true`, Flash affiche le support à sa taille préférée, sauf si la zone de lecture du support est inférieure à la taille préférée. Si tel est le cas, Flash réduit le support pour qu'il puisse s'adapter à la taille de l'occurrence et respecter les proportions. Si la taille préférée est inférieure à la zone de support de l'occurrence, une partie de la zone du support n'est pas utilisée.
- Si vous définissez cette propriété sur `false`, Flash utilise la taille de l'occurrence dans la mesure du possible, tout en respectant les proportions. Si les propriétés `Media.autoSize` et `Media.aspectRatio` sont toutes deux définies sur `false`, la zone du support du composant est remplie. Cette zone est située au-dessus des contrôles (dans la disposition par défaut), moins une marge de 8 pixels autour correspondant aux bords du composant.

La valeur par défaut est `true`.

Exemple

L'exemple suivant indique que le contrôle n'est pas lu en fonction de la taille de son support :

```
myMedia.autoSize = false;
```

Voir aussi

`Media.aspectRatio`

Media.backgroundStyle

S'applique à

MediaController

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

myMedia`.backgroundStyle`

Description

Propriété : valeur booléenne qui indique l'arrière-plan dessiné pour l'occurrence MediaController. Une valeur « `default` » indique que l'arrière-plan chromé est dessiné et une valeur « `none` » indique qu'aucun arrière-plan chromé n'est dessiné. La valeur par défaut est « `default` ».

Il ne s'agit pas d'une propriété de style et elle n'est donc pas affectée par les paramètres de style.

Exemple

L'exemple suivant indique que l'arrière-plan chromé n'est pas dessiné pour le contrôle :

```
myMedia.backgroundStyle = "none";
```

Media.bytesLoaded

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

myMedia.bytesLoaded

Description

Propriété en lecture seule : le nombre d'octets déjà chargés dans le composant pouvant être lus. La valeur par défaut est `undefined`.

Exemple

Le code suivant crée une variable appelée `PlaybackLoad` qui est définie avec le nombre d'octets chargés. Elle est ensuite utilisée dans une boucle `for`.

```
// Création d'une variable qui contient le nombre d'octets chargés.
var PlaybackLoad:Number = myMedia.bytesLoaded;
// Exécution d'une fonction jusqu'à ce que la lecture soit prête.
for (PlaybackLoad < 150) {
  someFunction();
}
```

Media.bytesTotal

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

myMedia.bytesTotal

Description

Propriété en lecture seule : le nombre d'octets à charger dans le composant MediaPlayback ou MediaDisplay. La valeur par défaut est `undefined`.

Exemple

L'exemple suivant indique à l'utilisateur la taille du support à diffuser en flux continu.

```
myTextField.text = myMedia.bytesTotal;
```

Media.change

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
listenerObject = new Object();
listenerObject.change = function(eventObject){
  // Insertion du code ici.
}
myMedia.addEventListener("change", listenerObject)
```

Description

Evénement : diffusé par les composants MediaDisplay et MediaPlayback pendant la lecture du support. Le pourcentage atteint peut être récupéré depuis l'occurrence de composant.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`eventObject`*) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. L'objet événement de l'événement `Media.change` possède deux propriétés supplémentaires :

`target` Référence à l'objet diffusé.

`type` Chaîne « `change` », qui indique le type d'événement.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

L'exemple suivant utilise un écouteur d'objet pour déterminer la position de la tête de lecture (`Media.playheadTime`), à partir de laquelle le pourcentage atteint peut être calculé :

```
var myPlayerListener:Object = new Object();
myPlayerListener.change = function(eventObj:Object) {
  var myPosition:Number = myPlayer.playheadTime;
  var myPercentPosition:Number = (myPosition/myPlayer.totalTime);
};
myPlayer.addEventListener("change", myPlayerListener);
```

Voir aussi

`Media.playing`, `Media.pause()`

Media.click

S'applique à

MediaController, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.click = function(eventObj:Object) {
  // ...
};
myMedia.addEventListener("click", listenerObject);
```

Description

Evénement : diffusé lorsque l'utilisateur clique sur le bouton Lire/Pause. Le champ detail peut être utilisé pour indiquer sur quel bouton l'utilisateur a cliqué. L'objet événement `Media.click` possède les propriétés suivantes :

`detail` Chaîne « `pause` » ou « `play` ».

`target` Référence à l'occurrence MediaController ou MediaPlayback.

`type` Chaîne « `click` ».

Exemple

Pour une occurrence du composant MediaController appelée **myMedia** (et avec un composant Window dans la bibliothèque), l'exemple suivant ouvre une fenêtre contextuelle lorsque l'utilisateur clique sur le bouton Lire/Pause :

```
var myMediaListener:Object = new Object();
myMediaListener.click = function(eventObj:Object) {
  mx.managers.PopUpManager.createPopUp(_root, mx.containers.Window, true);
};
myMedia.addEventListener("click", myMediaListener);
```

Media.complete

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.complete = function(eventObj:Object) {
  // ...
};
myMedia.addEventListener("complete", listenerObject);
```

Description

Evénement : notification indiquant que la tête de lecture a atteint la fin du support. L'objet événement `Media.complete` possède les propriétés suivantes :

`target` Référence à l'occurrence MediaDisplay ou MediaPlayback.

`type` Chaîne « `complete` ».

Exemple

L'exemple suivant utilise un écouteur d'objet pour indiquer à quel moment la lecture du support est terminée :

```
var myListener:Object = new Object();
myListener.complete = function(eventObj:Object) {
    trace("media is Finished");
};
myMedia.addEventListener("complete", myListener);
```

Media.contentPath

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
myMedia.contentPath
```

Description

Propriété : une chaîne contenant le chemin relatif et le nom de fichier du support devant être diffusé en flux continu et/ou lu. La définition de la propriété `contentPath` équivaut à appeler la méthode `Media.setMedia()` sans spécifier de paramètre *`mediaType`*. Si aucun paramètre *`mediaType`* n'est défini avec `Media.setMedia()`, le type par défaut est FLV. La valeur par défaut de la propriété `contentPath` est `undefined`.

Exemple

L'exemple suivant affiche le nom du support lu dans une zone de texte :

```
myTextField.text = myMedia.contentPath;
```

Voir aussi

`Media.setMedia()`

Media.controllerPolicy

S'applique à

MediaController, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
myMedia.controllerPolicy
```

Description

Propriété ; détermine si le composant MediaController (ou le sous-composant du contrôleur dans le composant MediaPlayback) est masqué lorsqu'il est instancié et apparaît uniquement lorsque l'utilisateur fait passer la souris sur l'état réduit du contrôleur.

Les valeurs possibles pour cette propriété sont les suivantes :

- `"on"` indique que les contrôles sont toujours développés.
- `"off"` indique que les contrôles sont toujours réduits.
- `"auto"` (valeur par défaut) indique que le contrôle reste à l'état réduit jusqu'à ce que l'utilisateur déplace la souris sur la zone active. La zone active correspond à la zone dans laquelle est dessiné le contrôle réduit. Le contrôle reste développé jusqu'à ce que la souris quitte cette zone.

REMARQUE	La zone active se développe et se réduit avec le contrôleur.

Exemple

L'exemple suivant maintient le contrôleur ouvert en permanence :

```
myMedia.controllerPolicy = "on";
```

Media.controlPlacement

S'applique à

MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
myMedia.controlPlacement
```

Description

Propriété : détermine l'endroit où la portion du contrôleur du composant MediaPlayback est positionnée par rapport à son affichage. Les valeurs possibles sont `"top"`, `"bottom"`, `"left"` et `"right"`. La valeur par défaut est `"bottom"`.

Exemple

Dans l'exemple suivant, la portion du contrôleur du composant MediaPlayback est située sur le côté droit :

```
myMedia.controlPlacement = "right";
```

Media.cuePoint

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObj:Object) {
  // ...
};
myMedia.addEventListener("cuePoint", listenerObject);
```

Description

Evénement : notification indiquant que la tête de lecture a atteint le point de repère. L'objet événement `Media.cuePoint` possède les propriétés suivantes :

`cuePointName` Chaîne indiquant le nom du point de repère.

`cuePointTime` Nombre, exprimé en images ou en secondes, qui indique le moment où le point de repère est atteint.

`target` Référence à l'objet MediaPlayback, le cas échéant, ou à l'objet MediaDisplay proprement dit.

`type` Chaîne « `cuePoint` ».

Exemple

L'exemple suivant utilise un écouteur d'objet pour déterminer à quel moment un point de repère est atteint :

```
var myCuePointListener:Object = new Object();
myCuePointListener.cuePoint = function(eventObject:Object){
  trace("heard " + eventObject.type + ", " + eventObject.target + ", " +
  eventObject.cuePointName + ", " + eventObject.cuePointTime);
};
myPlayback.addEventListener("cuePoint", myCuePointListener);
```

Voir aussi

`Media.addCuePoint()`, `Media.cuePoints`, `Media.getCuePoint()`

Media.cuePoints

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`myMedia``.cuePoints`

ou

`myMedia``.cuePoints[`*`N`*`]`

Description

Propriété : un tableau d'objets point de repère ayant été affectés à une occurrence de MediaPlayback ou de MediaDisplay. Dans le tableau, chaque objet point de repère peut avoir un nom, un temps en secondes ou en images et une propriété de lecteur (qui est le nom d'occurrence du composant auquel il est associé). La valeur par défaut est un tableau vide (`[]`).

Exemple

L'exemple suivant supprime le troisième point de repère en cas de lecture d'aperçu d'une action :

```
if (myVariable == actionPreview) {
  myMedia.removeCuePoint(myMedia.cuePoints[2]);
}
```

Voir aussi

`Media.addCuePoint()`, `Media.getCuePoint()`, `Media.removeCuePoint()`

Media.displayFull()

S'applique à

MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`myMedia``.displayFull()`

Valeur renvoyée

Aucune.

Description

Méthode : définit l'occurrence de MediaPlayback en mode plein écran. Dans ce mode, le composant se développe pour remplir la totalité de la scène. Pour que le composant retrouve sa taille normale, utilisez `Media.displayNormal()`.

Exemple

Le code suivant force le composant à se développer pour s'adapter à la taille de la scène :

```
myMedia.displayFull();
```

Voir aussi

`Media.displayNormal()`

Media.displayNormal()

S'applique à

MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

myMedia`.displayNormal()`

Valeur renvoyée

Aucune.

Description

Méthode : restaure l'occurrence MediaPlayback à sa taille normale après l'utilisation d'une méthode `Media.displayFull()`.

Exemple

Le code suivant renvoie un composant MediaPlayback à sa taille originale :

```
myMedia.displayNormal();
```

Voir aussi

`Media.displayFull()`

Media.getCuePoint()

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

myMedia.getCuePoint(*cuePointName*)

Paramètres

cuePointName Chaîne fournie lors de l'utilisation de `Media.addCuePoint()`.

Valeur renvoyée

Un objet point de repère.

Description

Méthode : renvoie un objet point de repère en fonction de son nom de point de repère.

Exemple

Le code suivant récupère un point de repère appelé `myCuePointName`.

```
myMedia.removeCuePoint(myMedia.getCuePoint("myCuePointName"));
```

Voir aussi

`Media.addCuePoint()`, `Media.cuePoint`, `Media.cuePoints`, `Media.removeCuePoint()`

Media.horizontal

S'applique à

MediaController

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

myMedia.horizontal

Description

Propriété : détermine si le composant MediaController est affiché verticalement ou horizontalement. Une valeur `true` indique que le composant est affiché horizontalement et une valeur `false` indique une orientation verticale. Lorsque la propriété est définie sur `false`, la barre de lecture et le curseur de lecture se déplacent de bas en haut. La valeur par défaut est `true`.

Exemple

L'exemple suivant affiche le composant MediaController dans une position verticale :

```
myMedia.horizontal = false;
```

Media.mediaType

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

myMedia.mediaType

Description

Propriété : indique le type de support (FLV ou MP3) à lire. La valeur par défaut est « `FLV` ». Voir « Utilisation de la vidéo » dans *Utilisation de Flash*.

Exemple

L'exemple suivant détermine le type de support actuel en cours de lecture :

```
var currentMedia:String = myMedia.mediaType;
```

Voir aussi

`Media.setMedia()`

Media.pause()

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
myMedia.pause()
```

Valeur renvoyée

Aucune.

Description

Méthode : met la tête de lecture en pause à l'emplacement actuel.

Exemple

Le code suivant met la lecture en pause.

```
myMedia.pause();
```

Media.play()

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
myMedia.play(startingPoint)
```

Paramètres

startingPoint Nombre entier positif indiquant le point de départ (en secondes) de la lecture du support.

Valeur renvoyée

Aucune.

Description

Méthode : lit le support associé à l'occurrence de composant à un point de départ donné. La valeur par défaut est la valeur actuelle de `playheadTime`.

Exemple

Le code suivant indique que la lecture du composant média doit commencer à 120 secondes :

```
myMedia.play(120);
```

Voir aussi

`Media.pause()`

Media.playheadChange

S'applique à

MediaController, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
listenerObject = new Object();
listenerObject.playheadChange = function(eventObject){
  // Insertion du code ici.
}
myMedia.addEventListener("playheadChange", listenerObject)
```

Description

Evénement : diffusé par le composant MediaController ou MediaPlayback lorsque l'utilisateur déplace le curseur de lecture ou clique sur le bouton Rembobiner ou Atteindre la fin. L'objet événement `Media.playheadChange` possède les propriétés suivantes :

`detail` Nombre indiquant le pourcentage de support qui a été lu.

`type` Chaîne « `"playheadChange` ».

Exemple

L'exemple suivant envoie le pourcentage lu vers le panneau Sortie lorsque l'utilisateur arrête de déplacer la tête de lecture :

```
var controlListen:Object = new Object();
controlListen.playheadChange = function(eventObj:Object) {
  trace(eventObject.detail);
};
myMedia.addEventListener("playheadChange", controlListen);
```

Media.playheadTime

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`myMedia``.playheadTime`

Description

Propriété : contient la position actuelle de la tête de lecture (en secondes) pour le scénario du support en cours de lecture. La valeur par défaut est l'emplacement de la tête de lecture.

Exemple

L'exemple suivant définit une variable à l'emplacement de la tête de lecture, indiquée en secondes :

```
var myPlayhead:Number = myMedia.playheadTime;
```

Media.playing

S'applique à

MediaDisplay, MediaPlayback, MediaController

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

myMedia.playing

Description

Propriété : renvoie une valeur booléenne indiquant si le support est en cours de lecture (`true`) ou en pause (`false`). Cette propriété est en lecture seule pour les composants MediaDisplay et MediaPlayback et a un accès en lecture/écriture pour le composant MediaController.

Exemple

Le code suivant détermine si le support est en cours de lecture ou en pause :

```
if(myMedia.playing == true){
  some function;
}
```

Voir aussi

Media.change

Media.preferredHeight

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

myMedia.preferredHeight

Description

Propriété ; définie en fonction de la valeur de la hauteur par défaut d'un fichier FLV. Cette propriété s'applique uniquement aux supports FLV car la hauteur des fichiers MP3 est fixe. Cette propriété peut être utilisée pour définir les propriétés de hauteur et de largeur (ainsi qu'une marge pour le composant lui-même). La valeur par défaut est `undefined` si aucun support FLV n'est défini.

Exemple

L'exemple suivant redimensionne une occurrence de MediaPlayback par rapport au support qu'elle lit et tient compte de la marge de pixels nécessaire à l'occurrence de composant :

```
if (myPlayback.contentPath != undefined) {
  var mediaHeight:Number = myPlayback.preferredHeight;
  var mediaWidth:Number = myPlayback.preferredWidth;
  myPlayback.setSize((mediaWidth + 20), (mediaHeight + 70));
}
```

Media.preferredWidth

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

myMedia.preferredWidth

Description

Propriété ; définie en fonction de la valeur de la largeur par défaut d'un fichier FLV. La valeur par défaut est `undefined`.

Exemple

L'exemple suivant définit la largeur souhaitée de la variable `mediaWidth` :

```
var mediaWidth:Number = myMedia.preferredWidth;
```

Media.progress

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.progress = function(eventObj:Object) {
  // ...
};
myMedia.addEventListener("progress", listenerObject);
```

Description

Evénement : est généré en continu jusqu'à la fin du téléchargement du support.
L'objet événement `Media.progress` possède les propriétés suivantes :

`target` Référence à l'occurrence MediaDisplay ou MediaPlayback.

`type` Chaîne « `progress` ».

Exemple

L'exemple suivant écoute la progression :

```
var myProgressListener:Object = new Object();
myProgressListener.progress = function(eventObj:Object) {
  // Exécution du clignotement lightMovieClip pendant la progression.
  var lightVisible:Boolean = lightMovieClip.visible;
  lightMovieClip.visible = !lightVisible;
};
```

L'exemple suivant écoute la progression et appelle une autre fonction si l'événement progress se poursuit pendant plus de 3 000 millisecondes (3 secondes) :

```
// Durée du délai avant d'appeler timeOut.
var timeOut:Number = 3000;

// Si timeOut a été atteint, action à effectuer :
function callback(arg) {
  trace(arg);
}

// Ecoute de la progression.
var myListener:Object = new Object();
myListener.progress = function(eventObj:Object) {
  setInterval(callback, timeOut, "Experiencing Network Delay");
};
md.addEventListener("progress", myListener);
```

Media.scrubbing

S'applique à

MediaController, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.progress = function(eventObj:Object) {
  // ...
};
myMedia.addEventListener("scrubbing", listenerObject);
```

Description

Evénement : généré lorsque l'utilisateur déplace la tête de lecture.

`target` Référence à l'occurrence MediaController ou MediaPlayback.

`type` Chaîne « `scrubbing` ».

Exemple

L'exemple suivant écoute l'utilisateur pour déplacer la tête de lecture :

```
my_mp.addEventListener("scrubbing", scrubbingListener);
function scrubbingListener(evt_obj:Object):Void {
  trace(evt_obj.type+" @ "+getTimer()+" ms
  (isScrubbing="+evt_obj.detail+")");
}
```

Media.removeAllCuePoints()

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
myMedia.removeAllCuePoints()
```

Valeur renvoyée

Aucune.

Description

Méthode : supprime tous les objets point de repère associés à une occurrence de composant.

Exemple

Le code suivant supprime tous les objets point de repère :

```
myMedia.removeAllCuePoints();
```

Voir aussi

`Media.addCuePoint()`, `Media.cuePoints`, `Media.removeCuePoint()`

Media.removeCuePoint()

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
myMedia.removeCuePoint(cuePoint)
```

Paramètres

cuePoint Référence à un objet point de repère ayant été précédemment affecté par le biais de `Media.addCuePoint()`.

Valeur renvoyée

Aucune.

Description

Méthode : supprime un point de repère associé à une occurrence de composant.

Exemple

Le code suivant supprime un point de repère appelé `myCuePoint` :

```
myMedia.removeCuePoint(getCuePoint("myCuePoint"));
```

Voir aussi

`Media.addCuePoint()`, `Media.cuePoints`, `Media.removeAllCuePoints()`

Media.setMedia()

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

myMedia.setMedia(*contentPath [, mediaType*])

Paramètres

contentPath Chaîne indiquant l'URL du support à lire. La valeur par défaut est `undefined`.

mediaType Chaîne utilisée pour définir le type de support sur FLV ou MP3. Ce paramètre est facultatif. La valeur par défaut est FLV.

Valeur renvoyée

Aucune.

Description

Méthode : définit le type de support et le chemin vers le type de support spécifié à l'aide d'un paramètre d'URL.

Cette méthode permet de définir le chemin du contenu et le type du support pour les composants MediaPlayback et MediaDisplay. La propriété `Media.contentPath` peut également être utilisée pour définir le chemin du contenu, mais elle ne permet pas de définir le type de support.

Si vous utilisez des fichiers FLV uniquement, il est inutile de définir un paramètre *mediaType*. Si vous utilisez des fichiers MP3 exclusivement, vous devez définir le paramètre *mediaType* sur MP3 une seule fois. Si vous passez sans cesse des fichiers FLV aux fichiers MP3, vous devez modifier chaque fois le type de support dans votre appel `setMedia()`. Si vous tentez de lire un fichier MP3 sans définir explicitement le type de support sur MP3, le fichier n'est pas lu.

Exemple

Le code suivant fournit de nouveaux supports pour lire une occurrence de composant :

```
myMedia.setMedia("http://www.helpexamples.com/flash/video/clouds.flv",
  "FLV");
```

Media.stop()

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
myMedia.stop()
```

Valeur renvoyée

Aucune.

Description

Méthode : arrête la tête de lecture et la déplace à la position 0, qui correspond au début du support.

Exemple

Le code suivant arrête la tête de lecture et la déplace à la position 0 :

```
myMedia.stop()
```

Media.totalTime

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
myMedia.totalTime
```

Description

Propriété : la durée totale du support, en secondes. Etant donné que le format de fichier FLV ne fournit pas sa durée de lecture à un composant média tant que son chargement n'est pas terminé, vous devez entrer manuellement `Media.totalTime` pour que la barre de lecture puisse refléter précisément la durée de lecture réelle du support. La valeur par défaut pour les fichiers MP3 est la durée de lecture du support. Pour les fichiers FLV, la valeur par défaut est `undefined`.

Vous ne pouvez pas définir cette propriété pour les fichiers MP3 car cette information est contenue dans l'objet Sound.

Exemple

L'exemple suivant définit la durée de lecture (en secondes) pour le support FLV :

```
myMedia.totalTime = 151;
```

Media.volume

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

myMedia`.volume`

Description

Propriété : stocke un nombre entier qui indique le paramètre volume, qui peut aller de 0 à 100. La valeur par défaut est 75.

Exemple

L'exemple suivant définit le volume maximum pour la lecture du support :

```
myMedia.volume = 100;
```

Voir aussi

`Media.volume`, `Media.pause()`

Media.volume

S'applique à

MediaController, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.volume = function(eventObj:Object) {
  // ...
};
myMedia.addEventListener("volume", listenerObject);
```

Description

Evénement : diffusé lorsque la valeur du volume est réglée par l'utilisateur. L'objet événement `Media.volume` possède les propriétés suivantes :

`detail` Nombre entier compris entre 0 et 100 qui représente le niveau de volume.

`type` Chaîne « `volume` ».

Exemple

L'exemple suivant informe l'utilisateur que le volume est en cours de réglage :

```
var myVolListener:Object = new Object();
myVolListener.volume = function(eventObj:Object) {
   mytextfield.text = "Volume adjusted!";
};
myMedia.addEventListener("volume", myVolListener);
```

Voir aussi

`Media.volume`

CHAPITRE 30

Composant Menu

30

Le composant Menu permet à un utilisateur de sélectionner un élément dans un menu contextuel, tout comme le menu Fichier ou Edition de la plupart des applications logicielles.

REMARQUE	Le composant Menu est pris en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Un composant Menu s'ouvre généralement dans une application lorsqu'un utilisateur survole un activateur de menu de type bouton ou clique sur ce dernier. Vous pouvez également écrire un script indiquant à un composant Menu de s'ouvrir lorsqu'un utilisateur appuie sur une touche spécifique.

Les composants Menu sont toujours créés dynamiquement lors de l'exécution. Pour créer un menu avec ActionScript, faites glisser le composant du panneau Composants vers la bibliothèque, puis utilisez le code suivant« »:

```
var myMenu = mx.controls.Menu.createMenu(parent, menuDataProvider);
```

Utilisez le code suivant pour ouvrir un menu dans une application :

```
myMenu.show(x, y);
```

Un événement `menuShow` est diffusé à l'ensemble des écouteurs des occurrences Menu avant que le menu ne soit rendu, afin de pouvoir mettre à jour l'état des éléments de menu. De la même façon, un événement `menuHide` est diffusé juste après le masquage d'une occurrence Menu.

Les éléments d'un menu sont décrits avec XML. Pour plus d'informations, voir « Présentation du composant Menu : affichage et données », à la page 922.

Vous ne pouvez pas rendre le composant Menu accessible aux lecteurs d'écran.

Les menus sont souvent imbriqués dans des barres de menus. Pour plus d'informations sur les barres de menus, reportez-vous à « Composant MenuBar », à la page 985.

Interaction avec le composant Menu

Vous pouvez utiliser la souris et le clavier pour interagir avec un composant Menu.

Une fois qu'un composant Menu est ouvert, il reste visible jusqu'à ce qu'il soit fermé par un script ou jusqu'à ce que l'utilisateur clique avec la souris en dehors du menu ou à l'intérieur d'un élément activé.

Vous pouvez sélectionner un élément de menu en cliquant sur ce dernier, excepté pour les types d'éléments de menu suivants :

Eléments désactivés ou séparateurs Les survols et clics n'ont aucune incidence (le menu reste visible).

Ancres d'un sous-menu Les survols activent le sous-menu, les clics n'ont aucune incidence. Le survol de tout élément autre que ceux appartenant au sous-menu ferme le sous-menu.

Lorsqu'un élément est sélectionné, un événement `Menu.change` est envoyé à l'ensemble des écouteurs du menu, le menu est masqué et les actions suivantes se produisent, en fonction du type d'élément :

case à cocher L'attribut `selected` de l'élément est activé.

bouton radio L'élément devient la sélection en cours de son groupe de boutons radio.

Le déplacement de la souris déclenche des événements `Menu.rollOut` et `Menu.rollOver`.

Le fait d'appuyer sur le bouton de la souris en dehors du menu ferme le menu et déclenche un événement `Menu.menuHide` .

Le fait de relâcher le bouton de la souris dans un élément activé affecte les types d'éléments comme suit :

case à cocher L'attribut `selected` de l'élément est activé.

bouton radio L'attribut `selected` de l'élément est défini sur `true` et l'attribut `selected` de l'élément précédemment sélectionné dans le groupe de boutons radio est défini sur `false`. La propriété `selection` de l'objet groupe de boutons radio correspondant est définie pour faire référence à l'élément de menu sélectionné.

undefined et le parent d'un menu hiérarchique La visibilité du menu hiérarchique est activée.

Lorsqu'une occurrence de Menu a le focus via un clic ou l'utilisation de la tabulation, vous pouvez utiliser les touches suivantes pour la contrôler :

Touche	Description
Flèche vers le bas Flèche vers le haut	Déplace la sélection de bas en haut dans les lignes du menu. La sélection passe dans la première ou la dernière ligne.
Flèche droite	Ouvre un sous-menu ou déplace la sélection vers le menu suivant dans une barre de menus (si une barre de menus est disponible).
Flèche gauche	Ferme un sous-menu et renvoie le focus au menu parent (si un menu parent est disponible) ou déplace la sélection vers le menu précédent dans une barre de menus (si une barre de menus est disponible).
Entrée	Ouvre un sous-menu. Si aucun sous-menu n'est disponible, cette touche a le même effet que si vous cliquez et relâchez le bouton de la souris sur une ligne.

REMARQUE

Si un menu est ouvert, vous pouvez appuyer sur la touche de tabulation pour quitter le menu. Vous devez effectuer une sélection ou quitter le menu en appuyant sur Echap.

Utilisation du composant Menu

Vous pouvez utiliser le composant Menu pour créer un menu d'options sélectionnables semblable au menu Fichier ou Edition de la plupart des applications logicielles. Vous pouvez également utiliser le composant Menu pour créer des menus contextuels qui s'affichent lorsque l'utilisateur clique sur une zone sensible ou appuie sur une touche de modification. Utilisez le composant Menu avec le composant MenuBar pour créer une barre de menus horizontale contenant des menus qui se développent sous chaque élément de la barre de menus.

Tout comme les menus d'ordinateur de bureau standard, le composant Menu prend en charge les éléments de menu dont les fonctions correspondent aux catégories générales suivantes :

Activateurs de commandes Ces éléments déclenchent des événements. Vous devez écrire du code pour gérer ces événements.

Ancres de sous-menus Ces éléments sont des ancres qui ouvrent des sous-menus.

Boutons radio Ces éléments fonctionnent par groupe. Vous ne pouvez sélectionner qu'un élément à la fois.

Eléments de case à cocher Ces éléments représentent une valeur booléenne (`true` ou `false`).

Séparateurs Ces éléments fournissent une ligne horizontale unique qui divise les éléments d'un menu en différents groupes visuels.

Présentation du composant Menu : affichage et données

Au niveau de sa conception, le composant Menu est constitué d'un modèle de données et d'une vue qui affiche les données. La classe Menu est la vue qui contient les méthodes de configuration visuelle. La Classe MenuDataProvider ajoute des méthodes à l'objet prototype XML global (tout comme l'API DataProvider le fait pour l'objet Array). Ces méthodes vous permettent de créer des fournisseurs de données de façon externe et de les ajouter à plusieurs occurrences de menu. Le fournisseur de données diffuse toutes les modifications à l'ensemble de ses vues client. (Voir la section « Classe MenuDataProvider », à la page 972).

Une occurrence de Menu est un ensemble hiérarchique d'éléments XML correspondant aux éléments de menu individuels. Les attributs définissent le comportement et l'apparence de l'élément de menu correspondant à l'écran. Cet ensemble peut facilement être traduit depuis et vers le langage XML, qui est utilisé pour décrire les menus (la balise `menu`) et les éléments (la balise `menuitem`). La classe XML ActionScript intégrée est la base du modèle sous-jacent du composant Menu.

Un menu simple contenant deux éléments peut être décrit dans XML avec deux sous-éléments d'élément de menu :

```
<menu>
  <menuitem label="Up" />
  <menuitem label="Down" />
</menu>
```

REMARQUE

Les noms de balise des nœuds XML (`menu` et `menuitem`) ne sont pas importants. Les attributs et leurs relations d'imbrication sont utilisés dans le menu.

Présentation des menus hiérarchiques

Pour créer des menus hiérarchiques, imbriquez des éléments XML dans un élément XML parent de la manière suivante :

```
<menu>
    <menuitem label="MenuItem A" >
      <menuitem label="SubMenuItem 1-A" />
      <menuitem label="SubMenuItem 2-A" />
    </menuitem>
    <menuitem label="MenuItem B" >
      <menuitem label="SubMenuItem 1-B" />
      <menuitem label="SubMenuItem 2-B" />
    </menuitem>
</menu>
```

Ceci convertit l'élément du menu parent en ancre de menu contextuel, pour qu'il ne génère pas d'événement une fois sélectionné.

Présentation des attributs XML d'élément de menu

Les attributs de l'élément XML d'un élément de menu déterminent ce qui est affiché, le comportement de l'élément de menu et la façon dont il est exposé à ActionScript. Le tableau suivant décrit les attributs d'un élément de menu XML :

Nom de l'attribut	Type	Valeur par défaut	Description
`label`	Chaîne	`undefined`	Le texte affiché pour représenter un élément de menu. Cet attribut est requis pour tous les types d'élément, excepté pour `separator`.
`type`	`separator`, `check`, `radio`, `normal` ou `undefined`	`undefined`	Le type d'élément de menu : `separator`, `check box`, `radio button` ou `normal` (une commande ou un activateur de sous-menu). Si cet attribut n'existe pas, la valeur par défaut est `normal`.
`icon`	String	`undefined`	L'identificateur de liaison d'un actif d'image. Cet attribut n'est pas requis ni disponible pour le type `check`, `radio` ou `separator`.
`instanceName`	String	`undefined`	Un identificateur que vous pouvez utiliser pour faire référence à l'occurrence d'élément de menu depuis l'occurrence de menu racine. Par exemple, un élément de menu nommé *yellow* peut être référencé en tant que `myMenu.yellow`. Cet attribut n'est pas requis.

Nom de l'attribut	Type	Valeur par défaut	Description
`groupName`	String	`undefined`	Un identifiant que vous pouvez utiliser pour associer plusieurs boutons radio au sein d'un groupe radio et pour exposer l'état d'un groupe radio à partir de l'occurrence de menu racine. Par exemple, un groupe radio nommé *colors* peut être référencé en tant que `myMenu.colors`. Cet attribut est uniquement requis pour le type `radio`.
`selected`	Une valeur booléenne (`false` ou `true`) ou une chaîne (`"false"` ou `"true"`)	`false`	Une valeur booléenne ou de type chaîne indiquant si un élément `check` ou `radio` est activé (`true`) ou désactivé (`false`). Cet attribut n'est pas requis.
`enabled`	Une valeur booléenne (`false` ou `true`) ou une chaîne (« `false` » ou « `true` »)	`true`	Une valeur booléenne ou de type chaîne indiquant si l'élément de menu peut être sélectionné (`true`) ou non (`false`). Cet attribut n'est pas requis.

Présentation des types d'élément de menu

Il existe quatre types d'élément de menu, spécifiés par l'attribut `type` :

```
<menu>
    <menuitem label="Normal Item" />
    <menuitem type="separator" />
    <menuitem label="Checkbox Item" type="check" instanceName="check_1"/>
    <menuitem label="RadioButton Item" type="radio"
  groupName="radioGroup_1" />
</menu>
```

Eléments de menu Normal

L'élément de menu `normal` ne possède pas d'attribut `type`, ce qui signifie que son attribut `type` prend par défaut la valeur `normal`. Les éléments Normal peuvent être des activateurs de commande ou des activateurs de sous-menu, selon qu'ils comportent des sous-éléments imbriqués ou non.

Eléments de menu Separator

Un élément de menu dont l'attribut `type` est défini sur `separator` sert de séparateur visuel dans le menu. Le code XML suivant crée les trois éléments de menu Haut, Milieu et Bas, ainsi que des séparateurs entre chacun d'entre eux :

```
<menu>
    <menuitem label="Top" />
    <menuitem type="separator" />
    <menuitem label="Middle" />
    <menuitem type="separator" />
    <menuitem label="Bottom" />
</menu>
```

Tous les éléments separator sont désactivés. Le fait de cliquer ou de passer la souris sur un séparateur n'a aucun effet.

Eléments de menu Check box

Un élément de menu dont l'attribut `type` est défini sur `check` sert de case à cocher dans le menu. Lorsque l'attribut `selected` est défini sur `true`, une coche apparaît en regard de l'étiquette de l'élément de menu. Lorsqu'un élément check box est sélectionné, son état change automatiquement et un événement `change` est diffusé à tous les écouteurs du menu racine. Néanmoins, lorsqu'un élément de menu check box se comporte de la même façon qu'un composant CheckBox, un élément de menu check box apparaît visuellement sans la case autour de la coche. Par conséquent, un élément de menu check box non sélectionné apparaît comme un élément de menu normal tant qu'il n'est pas sélectionné.

L'exemple suivant définit trois éléments de menu de type check box :

```
<menu>
    <menuitem label="Apples" type="check" instanceName="buyApples"
  selected="true" />
    <menuitem label="Oranges" type="check" instanceName="buyOranges"
  selected="false" />
    <menuitem label="Bananas" type="check" instanceName="buyBananas"
  selected="false" />
</menu>
```

Dans ActionScript, les noms d'occurrence vous permettent d'accéder directement aux éléments de menu à partir du menu, comme dans l'exemple suivant :

```
myMenu.setMenuItemSelected(myMenu.buyapples, true);
myMenu.setMenuItemSelected(myMenu.buyoranges, false);
```

REMARQUE

L'attribut `selected` doit être modifié uniquement à l'aide de la méthode `setMenuItemSelected()`. Vous pouvez examiner directement l'attribut `selected`, mais celui-ci renvoie une valeur de type chaîne `true` ou `false`.

Eléments de menu Radio button

Les éléments de menu dont l'attribut `type` est défini sur `radio` peuvent être groupés pour que seul un élément puisse être sélectionné à la fois. Lorsqu'un élément de menu radio button se comporte de la même façon qu'un composant RadioButton, un élément de menu radio button apparaît visuellement sans la bordure autour du bouton. Par conséquent, un élément de menu radio button non sélectionné apparaît comme un élément de menu Normal tant qu'il n'est pas sélectionné.

Pour créer un groupe radio, affectez la même valeur à l'attribut `groupName` de tous les éléments de menu à inclure dans le groupe, comme dans l'exemple suivant :

```
<menu>
    <menuitem label="Center" type="radio" groupName="alignment_group"
       instanceName="center_item"/>
    <menuitem type="separator" />
    <menuitem label="Top"    type="radio" groupName="alignment_group" />
    <menuitem label="Bottom" type="radio" groupName="alignment_group" />
    <menuitem label="Right"  type="radio" groupName="alignment_group" />
    <menuitem label="Left"   type="radio" groupName="alignment_group" />
</menu>
```

Lorsque l'utilisateur sélectionne l'un des éléments, la sélection en cours change automatiquement et un événement `change` est diffusé à tous les écouteurs du menu racine. Dans ActionScript, la propriété `selection` permet de faire référence à l'élément actuellement sélectionné dans le groupe radio, comme dans l'exemple suivant :

```
var selectedMenuItem = myMenu.alignment_group.selection;
myMenu.alignment_group = myMenu.center_item;
```

Chaque valeur `groupName` doit être unique dans le domaine de l'occurrence de menu racine.

REMARQUE

L'attribut `selected` doit être modifié uniquement à l'aide de la méthode `setMenuItemSelected()`. Vous pouvez examiner directement l'attribut `selected`, mais celui-ci renvoie une valeur de type chaîne `true` ou `false`.

Exposition des éléments de menu à ActionScript

Vous pouvez utiliser l'attribut `instanceName` pour affecter un identifiant unique à chaque élément de menu et ainsi pouvoir y accéder directement à partir du menu racine. Par exemple, le code XML suivant définit les attributs `instanceName` de chaque élément de menu :

```
<menu>
    <menuitem label="Item 1" instanceName="item_1" />
    <menuitem label="Item 2" instanceName="item_2" >
      <menuitem label="SubItem A" instanceName="sub_item_A" />
      <menuitem label="SubItem B" instanceName="sub_item_B" />
    </menuitem>
</menu>
```

Vous pouvez utiliser ActionScript pour accéder aux occurrences correspondantes et à leurs attributs directement depuis le composant menu, comme dans l'exemple suivant :

```
var aMenuItem = myMenu.item_1;
myMenu.setMenuItemEnabled(item_2, true);
var aLabel = myMenu.sub_item_A.attributes.label;
```

REMARQUE

Chaque attribut `instanceName` doit être unique dans le domaine de l'occurrence de menu racine (y compris dans tous les sous-menus du menu racine).

A propos des propriétés d'objet d'initialisation

Le paramètre *`initObject`* (objet d'initialisation) joue un rôle fondamental dans la création de la disposition du composant Menu. Ce paramètre est un objet possédant des propriétés. Chaque propriété représente l'un des attributs XML possibles d'un élément de menu (Pour obtenir la description des propriétés autorisées dans le paramètre *`initObject`*, reportez-vous à « Présentation des attributs XML d'élément de menu », à la page 923).

Le paramètre *`initObject`* est utilisé dans les méthodes suivantes :

- `Menu.addMenuItem()`
- `Menu.addMenuItemAt()`
- `MenuDataProvider.addMenuItem()`
- `MenuDataProvider.addMenuItemAt()`

L'exemple suivant crée un paramètre *initObject* avec deux propriétés, `label` et `instanceName` :

```
var i = myMenu.addMenuItem({label:"myMenuItem",
  instanceName:"myFirstItem"});
```

Vous pouvez conjuguer plusieurs propriétés pour créer un type particulier d'élément de menu. Vous affectez des propriétés spécifiques pour créer certains types d'éléments de menu (normal, séparateur, case à cocher ou bouton radio).

Pour initialiser un élément de menu normal, par exemple, utilisez le paramètre *initObject* suivant :

```
myMenu.addMenuItem({label:"myMenuItem", enabled:true, icon:"myIcon",
  instanceName:"myFirstItem"});
```

Pour initialiser un élément de menu separator, utilisez le paramètre *initObject* suivant :

```
myMenu.addMenuItem({type:"separator"});
```

Pour initialiser un élément de menu check box, utilisez le paramètre *initObject* suivant :

```
myMenu.addMenuItem({type:"check", label:"myMenuCheck", enabled:false,
  selected:true, instanceName:"myFirstCheckItem"})
```

Pour initialiser un élément de menu radio button, utilisez le paramètre *initObject*suivant :

```
myMenu.addMenuItem({type:"radio", label:"myMenuRadio1", enabled:true,
  selected:false, groupName:"myRadioGroup",
  instanceName:"myFirstRadioItem"})
```

Vous devez considérer les attributs `instanceName`, `groupName` et `type` d'un élément de menu comme des attributs en lecture seule. Définissez-les uniquement au moment où vous créez un élément (par exemple, dans un appel à `addMenuItem()`). Si vous modifiez ces attributs par la suite, ils risquent de produire des résultats imprévisibles.

Paramètres de menu

Vous pouvez définir les paramètres de création suivants pour chaque occurrence du composant Menu dans l'inspecteur Propriétés :

rowHeight indique la hauteur de chaque ligne, en pixels. La modification de la taille de la police n'affecte pas la hauteur de la ligne. La valeur par défaut est 20.

Vous pouvez rédiger du code ActionScript pour contrôler le composant Menu à l'aide de ses propriétés, méthodes et événements. Pour plus d'informations, voir « Menu class », à la page 938.

Création d'une application avec le composant Menu

Dans l'exemple ci-dessous, un développeur crée une application et utilise le composant Menu pour exposer quelques-unes des commandes que les utilisateurs peuvent émettre, telles que Ouvrir, Fermer et Enregistrer.

Pour créer une application avec le composant Menu :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser le composant Menu du panneau Composants vers la bibliothèque.

   Les menus sont créés dynamiquement à l'aide d'ActionScript.
3. Faites glisser un composant Button du panneau Composants vers la bibliothèque.

   Le bouton sera utilisé pour activer le menu.
4. Dans le panneau Actions, sur la première image, entrez le code suivant pour ajouter un écouteur d'événement pour écouter les événements `click` sur le bouton. Le code écoute également un événement `change` sur le menu et affiche le nom de l'élément de menu sélectionné dans le panneau Sortie :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
  - Composant Button dans la bibliothèque
*/

import mx.controls.Button;
import mx.controls.Menu;

this.createClassObject(Button, "menu_button", 10, {label:"Launch
  Menu"});

// Création d'un menu.
var my_menu:Menu = Menu.createMenu();

// Ajout d'éléments de menu.
my_menu.addMenuItem("Open");
my_menu.addMenuItem("Close");
my_menu.addMenuItem("Save");
my_menu.addMenuItem("Revert");

// Ajout d'un écouteur d'événement change pour détecter l'élément
// de menu sélectionné.
var menuListener:Object = new Object();
menuListener.change = function(evt_obj:Object) {
 var item_obj:Object = evt_obj.menuItem;
 trace("Item selected: "+item_obj.attributes.label);
};
```

```
my_menu.addEventListener("change", menuListener);

// Ajout d'un écouteur de bouton qui affiche le menu lorsque vous
// cliquez sur le bouton.
var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object) {
 var my_button:Button = evt_obj.target;
 // Affichage du menu en bas du bouton.
 my_menu.show(my_button.x, my_button.y + my_button.height);
};
menu_button.addEventListener("click", buttonListener);
```

5. Choisissez Contrôle > Tester l'animation.

   Cliquez sur le bouton Launch Menu (Lancer le menu) pour afficher le menu. Lorsque vous choisissez un élément de menu, une instruction `trace()` signale la sélection dans le panneau Sortie.

Pour utiliser les données XML d'un serveur afin de créer et de renseigner un menu :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser le composant Menu du panneau Composants jusqu'à la bibliothèque.

   Les menus sont créés dynamiquement à l'aide d'ActionScript.
3. Dans le panneau Actions, ajoutez le code suivant à la première image pour créer un menu, puis utilisez la propriété `dataProvider` pour charger des éléments de menu à partir d'une page Web :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

var my_menu:Menu = Menu.createMenu();

// Importation d'un fichier XML.
var myDP_xml:XML = new XML();
myDP_xml.ignoreWhite = true;
myDP_xml.onLoad = function(success:Boolean) {
 // Transmission des données au menu lorsqu'elles arrivent.
 if (success) {
  my_menu.dataProvider = myDP_xml.firstChild;
 }
};
myDP_xml.load("http://www.flash-mx.com/mm/xml/menu.xml");

// Affichage et positionnement des menus.
my_menu.show(100, 20);
```

REMARQUE	Les éléments de menu sont décrits par les enfants du premier enfant du document XML.

4. Sélectionnez Contrôle > Tester l'animation.

   La définition du menu xml de la page Web est fournie ici à titre de référence :

```
<?xml version="1.0" ?>
<menu>
<menuitem label="Undo" />
<menuitem type="separator" />
<menuitem label="Cut" />
<menuitem label="Copy" />
<menuitem label="Paste" />
<menuitem label="Clear" />
<menuitem type="separator" />
<menuitem label="Select All" />
</menu>
```

Pour utiliser une chaîne XML bien construite afin de créer et de renseigner un menu :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser le composant Menu du panneau Composants jusqu'à la bibliothèque.

   Les menus sont créés dynamiquement à l'aide d'ActionScript.
3. Dans le panneau Actions, ajoutez le code suivant à la première image pour créer un menu et lui ajouter quelques éléments :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création d'un objet XML qui servira de jeu préconfiguré.
var my_xml:XML = new XML();

// L'élément suivant créé n'apparaît pas dans le menu.
// L'appel à la méthode createMenu() (ci-dessous) attend la
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu. Il s'agit simplement d'une méthode
// pour créer facilement cet
// élément racine et lui donner un nom.
var theMenuElement_obj:Object = my_xml.addMenuItem("XXXXX");
```

```
// Ajout des éléments de menu.
theMenuElement_obj.addMenuItem({label:"Undo"});
theMenuElement_obj.addMenuItem({type:"separator"});
theMenuElement_obj.addMenuItem({label:"Cut"});
theMenuElement_obj.addMenuItem({label:"Copy"});
theMenuElement_obj.addMenuItem({label:"Paste"});
theMenuElement_obj.addMenuItem({label:"Clear", enabled:"false"});
theMenuElement_obj.addMenuItem({type:"separator"});
theMenuElement_obj.addMenuItem({label:"Select All"});

// Création de l'objet Menu.
var my_menu:Menu = Menu.createMenu(this, theMenuElement_obj);

// Affichage et positionnement des menus.
my_menu.show(100, 20);
```

4. Choisissez Contrôle > Tester l'animation.

Pour utiliser la classe MenuDataProvider afin de créer et de renseigner un menu :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser le composant Menu du panneau Composants jusqu'à la bibliothèque.

   Les menus sont créés dynamiquement à l'aide d'ActionScript.
3. Dans le panneau Actions, ajoutez le code suivant à la première image pour créer un menu et lui ajouter quelques éléments :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création d'un objet XML qui servira de jeu préconfiguré.
var xml = new XML();

// L'élément suivant créé n'apparaît pas dans le menu.
// L'appel à la méthode createMenu() (ci-dessous) attend la
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu. Il s'agit simplement d'une méthode
// pour créer facilement cet
// élément racine et lui donner un nom.
var theMenuElement = xml.addMenuItem("XXXXX");

// Ajout des éléments de menu.
theMenuElement.addMenuItem({label:"Undo"});
theMenuElement.addMenuItem({type:"separator"});
theMenuElement.addMenuItem({label:"Cut"});
```

```
theMenuElement.addMenuItem({label:"Copy"});
theMenuElement.addMenuItem({label:"Paste"});
theMenuElement.addMenuItem({label:"Clear", enabled:"false"});
theMenuElement.addMenuItem({type:"separator"});
theMenuElement.addMenuItem({label:"Select All"});
// Création de l'objet Menu.
var my_menu = mx.controls.Menu.createMenu(_root, theMenuElement);

my_menu.show(100, 20);
```

4. Choisissez Contrôle > Tester l'animation.

Personnalisation du composant Menu

La taille horizontale du menu s'ajuste pour contenir le texte le plus long. Vous pouvez également appeler la méthode `setSize()` pour définir la taille du composant. La taille des icônes ne doit pas dépasser 16 pixels par 16 pixels.

Utilisation de styles avec le composant Menu

Vous pouvez appeler la méthode `setStyle()` pour modifier le style du menu, de ses éléments et de ses sous-menus. Le composant Menu prend en charge les styles suivants :

Style	n	Description
`themeColor`	Halo	Couleur de base d'un composant. « `haloGreen` », « `haloBlue` » et « `haloOrange` » sont des valeurs possibles. La valeur par défaut est « `haloGreen` ».
`alternatingRowColors`	Les deux	Spécifie des couleurs alternées pour les lignes. La valeur peut être un tableau de couleurs, 0xFF00FF, 0xCC6699 et 0x996699, par exemple. Contrairement aux styles de couleur à valeur simple, `alternatingRowColors` n'accepte pas les noms de couleur. Les valeurs doivent être des codes de couleur numériques. Par défaut, ce style n'est pas défini, et `backgroundColor` est utilisé à sa place pour toutes les lignes.
`backgroundColor`	Les deux	Couleur d'arrière-plan du menu. La couleur par défaut est le blanc et est définie sur la déclaration de style de classe. Ce style est ignoré si `alternatingRowColors` est spécifié.

Style	n	Description
backgroundDisabledColor	Les deux	Couleur d'arrière-plan lorsque la propriété enabled du composant est définie sur « false ». La valeur par défaut est 0xDDDDDD (gris clair).
borderStyle	Les deux	Le composant Menu utilise une occurrence de RectBorder en tant que bordure et répond aux styles définis pour cette classe. Voir « Classe RectBorder », à la page 1111. Le style de bordure par défaut est « menuBorder ».
color	Les deux	Couleur du texte.
disabledColor	Les deux	Couleur du texte lorsque le composant est désactivé. La couleur par défaut est 0x848384 (gris foncé).
embedFonts	Les deux	Valeur booléenne qui indique si la police spécifiée dans fontFamily est de type intégré. Ce style doit être défini sur true si fontFamily fait référence à une police intégrée. Dans le cas contraire, la police intégrée n'est pas utilisée. Si ce style est défini sur true et que fontFamily ne fait pas référence à une police intégrée, aucun texte n'est affiché. La valeur par défaut est false.
fontFamily	Les deux	Nom de la police du texte. La valeur par défaut est « _sans ».
fontSize	Les deux	Taille, en points, de la police. La valeur par défaut est 10.
fontStyle	Les deux	Style de police : "normal" ou "italic". La valeur par défaut est "normal".
fontWeight	Les deux	Epaisseur de la police : "none" ou "bold". La valeur par défaut est "none". Tous les composants peuvent également accepter la valeur "normal" au lieu de "none" pendant un appel à la méthode setStyle(), mais les appels suivants à getStyle() renvoient "none".
textAlign	Les deux	Alignement du texte : "left", "right" ou "center". La valeur par défaut est "left".

Style	n	Description
`textDecoration`	Les deux	Décoration du texte : `"none"` ou `"underline"`. La valeur par défaut est`"none"`.
`textIndent`	Les deux	Nombre indiquant le retrait du texte. La valeur par défaut est 0.
`defaultIcon`	Les deux	Nom de l'icône par défaut à afficher sur chaque ligne. La valeur par défaut est `undefined`, ce qui signifie qu'aucune icône n'est affichée. La propriété icon n'est pas obligatoire, ne fonctionne pas pour les éléments « check », « radio » ou « separator » et utilise l'identificateur de liaison d'un actif d'image comme paramètre value. Tous les éléments de menu affichent la même icône.
`popupDuration`	Les deux	Durée de la transition lorsqu'un menu s'ouvre. La valeur est exprimée en millisecondes. 0 indique qu'il n'existe aucune transition. La valeur par défaut est 150.
`rollOverColor`	Les deux	Couleur d'arrière-plan d'une ligne survolée. La couleur par défaut est 0xE3FFD6 (vert vif) pour le thème Halo et 0xAAAAAA (gris clair) pour le thème Sample. Lorsque `themeColor` est modifié au moyen d'un appel setStyle(), la structure définit `rollOverColor` sur une valeur liée au `themeColor` choisi.
`selectionColor`	Les deux	Couleur d'arrière-plan d'une ligne sélectionnée. La valeur par défaut est 0xCDFFC1 (vert clair) avec le thème Halo et 0xEEEEEE (gris très clair) avec le thème Sample. Lorsque `themeColor` est modifié au moyen d'un appel setStyle(), la structure définit `selectionColor` sur une valeur liée au `themeColor` choisi.
`selectionDuration`	Les deux	Longueur de la transition d'un état normal à un état sélectionné, en millisecondes. La valeur par défaut est 200.

Style	n	Description
selectionEasing	Les deux	Référence à l'équation d'accélération utilisée pour contrôler la transition entre des états de sélection. L'équation par défaut utilise une formule sine in/out. Pour plus d'informations, consultez « Personnalisation des animations de composants » dans *Utilisation des composants ActionScript 2.0*.
textRollOverColor	Les deux	Couleur du texte lorsque le pointeur passe dessus. La valeur par défaut est 0x2B333C (gris foncé). Ce style est important lorsque vous définissez rollOverColor, car les deux paramètres doivent se compléter pour que ce texte soit facilement visualisable pendant le survol.
textSelectedColor	Les deux	Couleur du texte dans la ligne sélectionnée. La valeur par défaut est 0x005F33 (gris foncé). Ce style est important lorsque vous définissez selectionColor, car les deux paramètres doivent se compléter pour que ce texte soit facilement visualisable lors de la sélection.
useRollOver	Les deux	Détermine si le survol d'une ligne active sa mise en surbrillance. La valeur par défaut est true.

Définition des styles pour tous les composants Menu dans un document

La classe Menu hérite de la classe ScrollSelectList. Les propriétés de style de niveau de classe par défaut sont définies sur la classe ScrollSelectList qui est partagée par tous les composants basés sur des listes. Vous pouvez définir de nouvelles valeurs de style par défaut directement sur cette classe, les nouveaux paramètres sont alors reflétés dans tous les composants concernés.

```
_global.styles.ScrollSelectList.setStyle("backgroundColor", 0xFF00AA);
```

Pour définir une propriété de style sur les composants Menu uniquement, vous pouvez créer une déclaration CSSStyleDeclaration et l'enregistrer dans _global.styles.Menu.

```
import mx.styles.CSSStyleDeclaration;
if (_global.styles.Menu == undefined) {
  _global.styles.Menu = new CSSStyleDeclaration();
}
_global.styles.Menu.setStyle("backgroundColor", 0xFF00AA);
```

Lors de la création d'une déclaration de style de niveau classe, toutes les valeurs par défaut fournies par la déclaration `ScrollSelectList` sont perdues. Ceci inclut `backgroundColor` qui est nécessaire pour prendre en charge les événements de souris. Pour créer une déclaration de style de niveau classe et conserver les valeurs par défaut, utilisez une boucle `for..in` pour copier les anciens paramètres dans la nouvelle déclaration.

```
var source = _global.styles.ScrollSelectList;
var target = _global.styles.Menu;
for (var style in source) {
  target.setStyle(style, source.getStyle(style));
}
```

Pour plus d'informations sur les styles de niveau classe, reportez-vous à « Définition de styles pour une classe de composants » dans *Utilisation des composants ActionScript 2.0*.

Utilisation d'enveloppes avec le composant Menu

Le composant Menu utilise une occurrence de RectBorder pour sa bordure (voir « Classe RectBorder », à la page 1111).

Le composant Menu comporte des éléments visuels pour les graphiques de branche, coche, point du bouton radio et séparateurs. Aucune enveloppe ne peut être appliquée à ces éléments de façon dynamique, mais ils peuvent être copiés du dossier Flash UI Components 2/Themes/MMDefault/Menu Assets/States dans les deux thèmes et modifiés selon les besoins. Les identificateurs de liaison ne peuvent pas être modifiés et toutes les occurrences de Menu doivent utiliser les mêmes symboles.

Pour créer des symboles de clip pour les éléments Menu :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Choisissez Fichier > Importer > Ouvrir une bibliothèque externe et sélectionnez le fichier HaloTheme.fla.

   Ce fichier est stocké dans le dossier Configuration au niveau de l'application. Pour connaître son emplacement exact dans votre système d'exploitation, reportez-vous à « Présentation des thèmes » dans *Utilisation des composants ActionScript 2.0*.
3. Dans le panneau Bibliothèque du thème, développez le dossier Flash UI Components 2/Themes/MMDefault, puis faites glisser le dossier Menu Assets jusqu'à la bibliothèque de votre document.
4. Développez le dossier Menu Assets/States dans la bibliothèque de votre document.
5. Ouvrez les symboles que vous souhaitez personnaliser pour la modification.

   Par exemple, ouvrez le symbole MenuCheckEnabled.

6. Personnalisez le symbole selon vos besoins.

   Par exemple, modifiez l'image pour qu'elle devienne un X au lieu d'une coche.
7. Répétez les étapes 6 et 7 pour tous les symboles devant être personnalisés.
8. Cliquez sur le bouton Précédent pour revenir au scénario principal.
9. Faites glisser le composant Menu du panneau Composants jusqu'à la bibliothèque du document actuel.

   Le composant Menu est ajouté à la bibliothèque, ce qui le rend disponible lors de l'exécution.
10. Ajoutez ActionScript au scénario principal pour créer une occurrence Menu lors de l'exécution :

```
var myMenu = mx.controls.Menu.createMenu();
myMenu.addMenuItem({label: "One", type: "check", selected: true});
myMenu.addMenuItem({label: "Two", type: "check"});
myMenu.addMenuItem({label: "Three", type: "check"});
myMenu.show(0, 0);
```

11. Choisissez Contrôle > Tester l'animation.

Menu class

Inheritance MovieClip > Classe UIObject > Classe UIComponent > View > ScrollView > ScrollSelectList > Menu

Nom de classe ActionScript mx.controls.Menu

Les méthodes et propriétés de la classe Menu vous permettent de créer et de modifier des menus lors de l'exécution.

La définition d'une propriété de la classe de menu avec ActionScript annule le paramètre du même nom défini dans l'inspecteur Propriétés ou des composants.

Toutes les classes de composants ont une propriété `version` qui correspond à une propriété de classe. Les propriétés de classe ne sont disponibles que pour la classe elle-même. La propriété `version` renvoie une chaîne qui indique la version du composant. Pour accéder à cette propriété, utilisez le code suivant :

```
trace(mx.controls.Menu.version);
```

REMARQUE

Le code `trace(myMenuInstance.version);` renvoie `undefined`.

Méthodes de la classe Menu

Le tableau suivant répertorie les méthodes de la classe Menu.

Méthode	Description
`Menu.addMenuItem()`	Ajoute un élément de menu au menu.
`Menu.addMenuItemAt()`	Ajoute un élément de menu à un endroit spécifique du menu.
`Menu.createMenu()`	Crée une occurrence de la classe Menu. Il s'agit d'une méthode statique.
`Menu.getMenuItemAt()`	Obtient une référence à un élément de menu, à un emplacement spécifié.
`Menu.hide()`	Ferme un menu.
`Menu.indexOf()`	Renvoie l'index d'un élément de menu donné.
`Menu.removeAll()`	Supprime tous les éléments d'un menu.
`Menu.removeMenuItem()`	Supprime l'élément de menu spécifié.
`Menu.removeMenuItemAt()`	Supprime un élément de menu à un emplacement spécifique d'un menu.
`Menu.setMenuItemEnabled()`	Indique si un élément de menu est activé (`true`) ou non (`false`).
`Menu.setMenuItemSelected()`	Indique si un élément de menu est sélectionné (`true`) ou non (`false`).
`Menu.show()`	Ouvre un menu à un emplacement spécifique ou à son emplacement précédent.

Méthodes héritées de la classe UIObject

Le tableau suivant répertorie les méthodes de la classe Menu héritées de la classe UIObject. Pour appeler ces méthodes à partir de l'objet Menu, utilisez le formulaire *`MenuInstance.methodName`*.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés ou des composants.

Méthode	Description
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
`UIObject.setSize()`	Redimensionne l'objet à la taille demandée.
`UIObject.setSkin()`	Définit une enveloppe dans l'objet.
`UIObject.setStyle()`	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant répertorie les méthodes de la classe Menu héritées de la classe UIComponent. Pour appeler ces méthodes à partir de l'objet Menu, utilisez le formulaire *`MenuInstance.methodName`*.

Méthode	Description
`UIComponent.getFocus()`	Renvoie une référence à l'objet ayant le focus.
`UIComponent.setFocus()`	Attribue le focus à l'occurrence de composant.

Propriétés de la classe Menu

Le tableau suivant présente la propriété de la classe Menu.

Propriété	Description
`Menu.dataProvider`	Source de données d'un menu.

Propriétés héritées de la classe UIObject

Le tableau suivant répertorie les propriétés de la classe Menu héritées de la classe UIObject. Pour accéder à ces propriétés à partir de l'objet Menu, utilisez le formulaire *`MenuInstance.propertyName`*.

Propriété	Description
`UIObject.bottom`	Position du bord inférieur de l'objet par rapport au bord inférieur de son parent. Lecture seule.
`UIObject.height`	Hauteur de l'objet, en pixels. Lecture seule.
`UIObject.left`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.right`	Position du bord droit de l'objet par rapport au bord droit de son parent. Lecture seule.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Position du bord supérieur de l'objet par rapport à son parent. Lecture seule.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Largeur de l'objet, en pixels. Lecture seule.
`UIObject.x`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.y`	Bord supérieur de l'objet, en pixels. Lecture seule.

Propriétés héritées de la classe UIComponent

Le tableau suivant répertorie les propriétés de la classe Menu héritées de la classe UIComponent. Pour accéder à ces propriétés à partir de l'objet Menu, utilisez le formulaire *`MenuInstance.propertyName`*.

Propriété	Description
`UIComponent.enabled`	Indique si le composant peut recevoir le focus et la saisie.
`UIComponent.tabIndex`	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Evénements de la classe Menu

Le tableau suivant répertorie les événements de la classe Menu.

Evénement	Description
`Menu.change`	Diffusé lorsqu'un utilisateur effectue une modification dans un menu.
`Menu.menuHide`	Diffusé lorsqu'un menu se ferme.
`Menu.menuShow`	Diffusé lorsqu'un menu s'ouvre.
`Menu.rollOut`	Diffusé lorsque le pointeur cesse de survoler un élément.
`Menu.rollOver`	Diffusé lorsque le pointeur survole un élément.

Evénements hérités de la classe UIObject

Le tableau suivant répertorie les événements de la classe Menu hérités de la classe UIObject.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant répertorie les événements de la classe Menu hérités de la classe UIComponent.

Evénement	Description
`UIComponent.focusIn`	Diffusé lorsqu'un objet reçoit le focus.
`UIComponent.focusOut`	Diffusé lorsqu'un objet perd le focus.
`UIComponent.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIComponent.keyUp`	Diffusé lorsqu'une touche est relâchée.

Menu.addMenuItem()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
menuInstance.addMenuItem(initObject)
```

Utilisation 2 :

```
menuInstance.addMenuItem(childMenuItem)
```

Paramètres

`initObject` Objet qui contient les propriétés d'initialisation des attributs d'un élément de menu. Voir « Présentation des attributs XML d'élément de menu », à la page 923.

`childMenuItem` Objet nœud XML.

Valeur renvoyée

Une référence au nœud XML ajouté.

Description

Méthode : le premier exemple d'utilisation ajoute un élément de menu à la fin du menu. L'élément de menu est construit à partir des valeurs fournies dans le paramètre *`initObject`*. Le second exemple d'utilisation ajoute un élément de menu (nœud XML prédéfini) sous la forme d'un objet XML à la fin du menu. Si le nœud ajouté existe déjà, il est supprimé de son emplacement précédent.

Exemple

L'exemple suivant crée deux menus, ajoutant pour commencer un élément de menu à chacun d'eux. L'exemple ajoute ensuite deux autres éléments au premier menu, appelle `addMenuItem()` pour ajouter le premier élément de menu en spécifiant ses attributs. Il ajoute ensuite le second élément de menu à l'aide du nœud de l'élément de menu prédéfini depuis le deuxième menu.

Vous devez d'abord faire glisser un composant Menu jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création des objets Menu.
var first_menu:Menu = Menu.createMenu();
first_menu.addMenuItem({label:"1st Item"});
var second_menu:Menu = Menu.createMenu();
second_menu.addMenuItem({label:"1st Item 2nd Menu"});

// Première méthode d'utilisation
first_menu.addMenuItem({label:"Radio Item", instanceName:"radioItem1",
  type:"radio", selected:false, enabled:true, groupName:"myRadioGroup"});

// Seconde méthode d'utilisation
first_menu.addMenuItem(second_menu.getMenuItemAt(0));

// Affichage du menu.
first_menu.show();
```

Menu.addMenuItemAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
menuInstance.addMenuItemAt(index, initObject)
```

Utilisation 2 :

```
menuInstance.addMenuItemAt(index, childMenuItem)
```

Paramètres

index Nombre entier indiquant la position de l'index (parmi les nœuds enfants) à laquelle l'élément est ajouté.

initObject Objet qui contient les propriétés d'initialisation des attributs d'un élément de menu. Voir « Présentation des attributs XML d'élément de menu », à la page 923.

childMenuItem Objet nœud XML.

Valeur renvoyée

Une référence au nœud XML ajouté.

Description

Méthode : le premier exemple d'utilisation ajoute un élément de menu (nœud enfant) à l'emplacement indiqué dans le menu. L'élément de menu est construit à partir des valeurs fournies dans le paramètre *initObject*. Le second exemple d'utilisation ajoute un élément de menu (nœud XML prédéfini) sous la forme d'un objet XML à un emplacement spécifié dans le menu. Si le nœud ajouté existe déjà, il est supprimé de son emplacement précédent.

Exemple

L'exemple suivant crée deux menus, ajoutant pour commencer un élément de menu à chacun d'eux. Il ajoute ensuite deux autres éléments de menu au premier menu, appelle `addMenuItemAt()` pour ajouter un élément de menu en deuxième position en spécifiant ses attributs. Pour finir, il ajoute un élément de menu en troisième position à l'aide du nœud de l'élément de menu prédéfini depuis le deuxième menu.

Vous devez d'abord faire glisser un composant Menu jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création des objets Menu.
var first_menu:Menu = Menu.createMenu();
first_menu.addMenuItem({label:"1st Item"});
var second_menu:Menu = Menu.createMenu();
second_menu.addMenuItem({label:"1st Item 2nd Menu"});

// Première méthode d'utilisation
first_menu.addMenuItemAt(1, {label:"Radio Item", instanceName:"radioItem1",
  type:"radio", selected:false, enabled:true, groupName:"myRadioGroup"});

// Seconde méthode d'utilisation
first_menu.addMenuItemAt(2, second_menu.getMenuItemAt(0));

// Affichage du menu.
first_menu.show();
```

Menu.change

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.change = function(eventObject:Object) {
  // Insertion du code ici.
};
menuInstance.addEventListener("change", listenerObject);
```

Utilisation 2 :

```
on (change) {
  // Insertion du code ici.
}
```

Description

Evénement : diffusé à l'ensemble des écouteurs enregistrés chaque fois qu'un utilisateur provoque un changement dans le menu.

Les composants utilisent un modèle dispatcher(diffuseur)-écouteur d'événement. Lorsqu'un composant Menu diffuse un événement `change`, l'événement est géré par une fonction (également appelée *gestionnaire*) associée à un objet écouteur (*`listenerObject`*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`eventObject`*) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. L'objet événement de l'événement `Menu.change` possède les propriétés supplémentaires suivantes :

- `menuBar` Référence à l'occurrence MenuBar qui est le parent du menu cible. Elle prend la valeur `undefined` si le menu cible ne fait pas partie d'une occurrence MenuBar.
- `menu` Référence à l'occurrence Menu contenant l'élément cible.
- `menuItem` Nœud XML représentant l'élément de menu sélectionné.
- `groupName` Chaîne indiquant le nom du groupe radio dont l'élément fait partie. Elle prend la valeur `undefined` si l'élément ne fait pas partie d'un groupe radio.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

L'exemple suivant crée un menu, `my_menu` et lui définit un écouteur d'événement, `menulistener`, qui écoute un événement change. Lorsqu'un utilisateur provoque un événement change en cliquant sur un élément de menu, l'exemple affiche son attribut label dans le panneau Sortie.

Vous devez d'abord faire glisser un composant Menu jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création d'un objet XML qui servira de jeu préconfiguré.
var my_xml:XML = new XML();

// L'élément suivant créé n'apparaît pas dans le menu.
// L'appel à la méthode createMenu() (ci-dessous) attend la
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu. Il s'agit simplement d'une méthode
// pour créer facilement cet élément racine et lui donner un nom.
var menuDP_obj:Object = my_xml.addMenuItem("Edit");

// Ajout des éléments de menu.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});

// Création de l'objet Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);
my_menu.show();

var menuListener:Object = new Object();
menuListener.change = function(evt_obj:Object) {
 trace("Menu item chosen: " + evt_obj.menuItem.attributes.label);
};
my_menu.addEventListener("change", menuListener);
```

Menu.createMenu()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
Menu.createMenu([parent [, mdp]])
```

Paramètres

parent Occurrence MovieClip. Le clip est le composant parent contenant la nouvelle occurrence de Menu. Ce paramètre est facultatif.

mdp Occurrence MenuDataProvider qui décrit cette occurrence Menu. Ce paramètre est facultatif.

Valeur renvoyée

Une référence à la nouvelle occurrence de menu.

Description

Méthode (statique) : crée une occurrence de Menu et l'associe (éventuellement) au parent indiqué ; la classe MenuDataProvider est spécifiée en tant que source de données des éléments de menu.

Si le paramètre *parent* est omis ou null, le Menu est associé au scénario `_root`.

Si le paramètre *mdp* est omis ou null, le menu ne contient aucun élément ; vous devez appeler `addMenu()` ou `setDataProvider()` pour le renseigner.

Exemple

L'exemple suivant crée un menu avec un sous-menu pour l'élément de menu Nouveau. Il crée le menu en créant un objet XML, `my_xml` et en lui ajoutant des éléments de menu avec des appels à `addMenuItem()`. Il crée ensuite le menu avec un appel à `createMenu()`, en transmettant l'objet XML comme fournisseur de données.

Vous devez d'abord faire glisser un composant Menu jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;
```

```
var my_xml:XML = new XML();
var newItem_obj:Object = my_xml.addMenuItem({label:"New"});

// Création d'autres éléments de sous-menu pour le menu principal.
my_xml.addMenuItem({label:"Open", instanceName:"miOpen"});
my_xml.addMenuItem({label:"Save", instanceName:"miSave"});
my_xml.addMenuItem({type:"separator"});
my_xml.addMenuItem({label:"Quit", instanceName:"miQuit"});

// Création d'éléments de sous-menu pour le sous-menu « Nouveau ».
newItem_obj.addMenuItem({label:"File..."});
newItem_obj.addMenuItem({label:"Project..."});
newItem_obj.addMenuItem({label:"Resource..."});

// Création d'un menu.
var my_menu:Menu = Menu.createMenu(myParent_mc, my_xml);
my_menu.show(100, 20);
```

Menu.dataProvider

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

menuInstance.dataProvider

Description

Propriété : la source de données des éléments d'un composant Menu.

`Menu.dataProvider` est un objet nœud XML. La définition de cette propriété remplace la source de données actuelle du menu.

La valeur par défaut est `undefined`.

REMARQUE	Toutes les occurrences de XML ou de XMLNode héritent automatiquement des méthodes et propriétés de la classe MenuDataProvider lorsqu'elles sont utilisées avec le composant Menu.

Exemple

L'exemple suivant crée un menu (`my_menu`), charge des éléments de menu à partir d'une page Web dans un objet XML, puis remplit le menu avec des éléments de menu en affectant des nœuds enfants de l'objet XML à la propriété `dataProvider` du menu (`my_menu.dataProvider`).

Vous devez d'abord faire glisser un composant Menu jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création de l'objet Menu.
var my_menu:Menu = Menu.createMenu();

var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean){
 if (success) {
  my_menu.dataProvider = my_xml.firstChild;
 }
}
my_xml.load("http://www.flash-mx.com/mm/xml/menu.xml");

// Affichage et positionnement des menus.
my_menu.show(100, 20);
```

Menu.getMenuItemAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`menuInstance``.getMenuItemAt(`*`index`*`)`

Paramètres

`index` Nombre entier spécifiant l'index du nœud dans le menu.

Valeur renvoyée

Une référence au nœud spécifié.

Description

Méthode : renvoie une référence au nœud enfant spécifié du menu.

Exemple

L'exemple suivant crée d'abord deux menus avec un seul élément de menu pour chacun d'eux. Il ajoute ensuite un deuxième élément de menu au premier menu en appelant la méthode `getmenuItemAt()` pour obtenir l'élément du deuxième menu et l'ajouter au premier.

Vous devez d'abord faire glisser un composant Menu jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création des objets Menu.
var first_menu:Menu = Menu.createMenu();
first_menu.addMenuItem({label:"1st Item"});
var second_menu:Menu = Menu.createMenu();
second_menu.addMenuItem({label:"1st Item 2nd Menu"});

// Ajout de l'élément depuis second_menu en deuxième position
// du premier menu.
first_menu.addMenuItemAt(1, second_menu.getMenuItemAt(0));

// Affichage du menu.
first_menu.show();
```

Menu.hide()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

menuInstance.hide()

Valeur renvoyée

Aucune.

Description

Méthode : ferme un menu.

Exemple

L'exemple suivant crée un bouton et un menu à deux éléments, puis affiche le menu pendant un intervalle de 2 000 millisecondes. A la fin de l'intervalle, la fonction `closeMenu()` appelle la méthode `menu.hide()` pour fermer le menu. Cliquez sur le bouton Reset Menu (Réinitialiser le menu) pour déclencher l'écouteur `resetMenu()` qui affiche de nouveau le menu et réinitialise l'intervalle.

Vous devez d'abord faire glisser un composant Menu et un composant Button jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
  - Composant Button dans la bibliothèque
*/

import mx.controls.Button;
import mx.controls.Menu;

this.createClassObject(Button, "my_button", 10, {label:"Reset Menu",
  _x:100, _y:50});

// Création d'un objet XML qui servira de jeu préconfiguré.
var my_xml:XML = new XML();

// L'élément suivant créé n'apparaît pas dans le menu.
// L'appel à la méthode createMenu() (ci-dessous) attend la
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu. Il s'agit simplement d'une méthode
// pour créer facilement cet élément racine et lui donner un nom.
var menuDP_obj:Object = my_xml.addMenuItem("Edit");

// Ajout des éléments de menu.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});

// Création de l'objet Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

my_menu.show(100, 100);
```

```
// Appel à closeMenu au bout de 2 000 millisecondes.
var interval_id:Number = setInterval(closeMenu, 2000, my_menu);
function closeMenu(the_menu:Menu):Void {
  the_menu.hide();
  clearInterval(interval_id);
}
// Ecouteur du clic du bouton ; affichage du menu et réinitialisation
// de l'intervalle.
function resetMenu(evt_obj:Object):Void {
  clearInterval(interval_id);
  my_menu.show(100, 100);
  interval_id = setInterval(closeMenu, 2000, my_menu);
}
my_button.addEventListener("click", resetMenu);
```

Voir aussi

`Menu.show()`

Menu.indexOf()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`menuInstance``.indexOf(`*`item`*`)`

Paramètres

`item` Référence à un nœud XML qui décrit un élément de menu.

Valeur renvoyée

Index de l'élément de menu spécifié ou valeur `undefined` si l'élément ne fait pas partie de ce menu.

Description

Méthode : renvoie l'index, dans cette occurrence de menu, de l'élément de menu spécifié.

Exemple

L'exemple suivant crée un menu à deux éléments d'un fournisseur de données XML, puis lui ajoute un troisième élément et enregistre la référence renvoyée par la méthode `addMenuItem()`. Il appelle ensuite la méthode `indexOf()` à l'aide de la référence pour obtenir l'index de l'élément et l'afficher dans le panneau Sortie.

Vous devez d'abord faire glisser un composant Menu et un composant Button jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création d'un objet XML qui servira de jeu préconfiguré.
var my_xml:XML = new XML();

// L'élément suivant créé n'apparaît pas dans le menu.
// L'appel à la méthode createMenu() (ci-dessous) attend la
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu. Il s'agit simplement d'une méthode
// pour créer facilement cet élément racine et lui donner un nom.
var menuDP_obj:Object = my_xml.addMenuItem("Edit");

// Ajout des éléments de menu.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});

// Création de l'objet Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);
var myItem_obj:Object = my_menu.addMenuItem({label:"That item"});

// Affichage et positionnement des menus.
my_menu.show(100, 20);

var myIndex_num:Number = my_menu.indexOf(myItem_obj);
trace("Index of 'That Item': " + myIndex_num);
```

Menu.menuHide

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.menuHide = function(eventObject:Object) {
  // Insertion du code ici.
};
menuInstance.addEventListener("menuHide", listenerObject);
```

Utilisation 2 :

```
on (menuHide) {
  // Insertion du code ici.
}
```

Description

Evénement : diffusé à l'ensemble des écouteurs enregistrés chaque fois qu'un menu se ferme.

Les composants utilisent un modèle dispatcher(diffuseur)-écouteur d'événement. Lorsqu'un composant Menu distribue un événement `menuHide`, l'événement est géré par une fonction (également appelée *gestionnaire*) associée à un objet écouteur (`listenerObject`) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez les noms du gestionnaire et de l'objet écouteur en tant que paramètres.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (`eventObject`) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. L'objet événement de l'événement `Menu.menuHide` possède deux propriétés supplémentaires :

- `menuBar` Référence à l'occurrence MenuBar qui est le parent du menu cible. Elle prend la valeur `undefined` si le menu cible ne fait pas partie d'une occurrence MenuBar.
- `menu` Référence à l'occurrence Menu masquée.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

L'exemple suivant crée un bouton et un menu à deux éléments. Lorsque l'utilisateur clique sur le bouton, un écouteur d'événement click affiche le menu. Lorsque l'utilisateur clique une deuxième fois, le menu est masqué et un écouteur de l'événement `menuHide`, `menuListener`, affiche « menu closed » (menu fermé) dans le panneau Sortie.

Vous devez d'abord faire glisser un composant Menu et un composant Button jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
  - Composant Button dans la bibliothèque
*/

import mx.controls.Button;
import mx.controls.Menu;

this.createClassObject(Button, "my_button", 10);

// Création d'un objet XML qui servira de jeu préconfiguré.
var my_xml:XML = new XML();

// L'élément suivant créé n'apparaît pas dans le menu.
// L'appel à la méthode createMenu() (ci-dessous) attend la
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu. Il s'agit simplement d'une méthode
// pour créer facilement cet élément racine et lui donner un nom.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Ajout des éléments de menu.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});

// Création de l'objet Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Ajout d'un bouton qui affiche le menu lorsque vous cliquez sur le bouton.
var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object) {
 // Obtention de la référence au bouton.
 var the_button:Button = evt_obj.target;
 // Affichage du menu en bas du bouton.
 my_menu.show(the_button.x, the_button.y + the_button.height);
};
my_button.addEventListener("click", buttonListener);
```

```
// Création d'un objet écouteur.
var menuListener:Object = new Object();
menuListener.menuHide = function(evt_obj:Object) {
 trace("Menu closed.");
};

// Ajout de l'écouteur.
my_menu.addEventListener("menuHide", menuListener);
```

Voir aussi

Menu.menuShow

Menu.menuShow

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.menuShow = function(eventObject:Object) {
  // Insertion du code ici.
};
menuInstance.addEventListener("menuShow", listenerObject);
```

Utilisation 2 :

```
on (menuShow) {
  // Insertion du code ici.
}
```

Description

Evénement : diffusé à l'ensemble des écouteurs enregistrés chaque fois qu'un menu s'ouvre. Tous les nœuds parent ouvrent leurs menus pour afficher leurs enfants.

Les composants utilisent un modèle dispatcher(diffuseur)-écouteur d'événement. Lorsqu'un composant Menu distribue un événement `menuShow`, l'événement est géré par une fonction (également appelée *gestionnaire*) associée à un objet écouteur (`listenerObject`) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez les noms du gestionnaire et de l'objet écouteur (Listener) en tant que paramètres.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. L'objet événement de l'événement `Menu.menuShow` possède les propriétés supplémentaires suivantes :

- `menuBar` Référence à l'occurrence MenuBar qui est le parent du menu cible. Elle prend la valeur `undefined` si le menu cible ne fait pas partie d'une occurrence MenuBar.
- `menu` Référence à l'occurrence Menu affichée.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

L'exemple suivant crée un bouton et un menu à deux éléments. Lorsque l'utilisateur clique sur le bouton, un écouteur d'événement click affiche le menu. Un écouteur de l'événement `menuShow`, `menuListener` affiche « Menu open » (Menu ouvert) dans le panneau Sortie.

Vous devez d'abord faire glisser un composant Menu et un composant Button jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
  - Composant Button dans la bibliothèque
*/

import mx.controls.Button;
import mx.controls.Menu;

this.createClassObject(Button, "my_button", 10);

// Création d'un objet XML qui servira de jeu préconfiguré.
var my_xml:XML = new XML();

// L'élément suivant créé n'apparaît pas dans le menu.
// L'appel à la méthode createMenu() (ci-dessous) attend la
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu. Il s'agit simplement d'une méthode
// pour créer facilement cet élément racine et lui donner un nom.
var menuDP_obj:Object = my_xml.addMenuItem("Edit");

// Ajout des éléments de menu.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});

// Création de l'objet Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);
```

```
// Ajout d'un bouton qui affiche le menu lorsque vous cliquez sur le bouton.
var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object) {
 // Obtention de la référence au bouton.
 var the_button:Button = evt_obj.target;
 // Affichage du menu en bas du bouton.
 my_menu.show(the_button.x, the_button.y + the_button.height);
};
my_button.addEventListener("click", buttonListener);

// Création d'un objet écouteur.
var menuListener:Object = new Object();
menuListener.menuShow = function(evt_obj:Object) {
 trace("Menu open.");
};

// Ajout de l'écouteur.
my_menu.addEventListener("menuShow", menuListener);
```

Voir aussi

Menu.menuHide

Menu.removeAll()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

menuInstance.removeAll()

Valeur renvoyée

Aucune.

Description

Méthode : supprime tous les éléments et actualise le menu.

Exemple

L'exemple suivant crée un menu à deux éléments, puis supprime tous les nœuds du menu après un intervalle de 2 secondes.

Vous devez d'abord faire glisser un composant Menu jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création d'un objet XML qui servira de jeu préconfiguré.
var my_xml:XML = new XML();

// L'élément suivant créé n'apparaît pas dans le menu.
// L'appel à la méthode createMenu() (ci-dessous) attend la
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu. Il s'agit simplement d'une méthode
// pour créer facilement cet élément racine et lui donner un nom.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Ajout des éléments de menu.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});

// Création de l'objet Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Affichage et positionnement des menus.
my_menu.show(100, 20);
var interval_id:Number = setInterval(remove, 2000, my_menu);
function remove(the_menu:Menu):Void {
  // Suppression de tous les éléments de menu.
  the_menu.removeAll();
  clearInterval(interval_id);
  the_menu.show(100, 20);
}
```

Menu.removeMenuItem()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

menuInstance.removeMenuItem()

Valeur renvoyée

Une référence à l'élément de menu renvoyé (nœud XML). La valeur est `undefined` s'il n'y a aucun élément à cette position.

Description

Méthode : supprime l'élément de menu spécifié ainsi que tous ses enfants et actualise le menu.

Exemple

L'exemple suivant crée un menu à trois éléments, puis définit un intervalle pour que le menu s'affiche pendant 2 secondes. A l'expiration de l'intervalle, l'exemple appelle la fonction `removeItem()` qui appelle la méthode `removeMenuItem()` pour supprimer le premier élément dans le menu et le réafficher.

Vous devez d'abord faire glisser un composant Menu jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création d'un objet XML qui servira de jeu préconfiguré.
var my_xml:XML = new XML();

// L'élément suivant créé n'apparaît pas dans le menu.
// L'appel à la méthode createMenu() (ci-dessous) attend la
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu. Il s'agit simplement d'une méthode
// pour créer facilement cet élément racine et lui donner un nom.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");
```

```
// Ajout des éléments de menu.
menuDP_obj.addMenuItem({label:"first Item"});
menuDP_obj.addMenuItem({label:"second Item"});
menuDP_obj.addMenuItem({label:"third Item"});

// Création de l'objet Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Affichage et positionnement du menu.
my_menu.show(100, 20);

// Appel à closeMenu au bout de 2 000 millisecondes.
var interval_id:Number = setInterval(removeItem, 2000, my_menu);
function removeItem(the_menu:Menu):Void {
  // Suppression du premier élément de nœud.
  var myItem_obj:Object = my_menu.getMenuItemAt(0);
  myItem_obj.removeMenuItem();
  clearInterval(interval_id);
  my_menu.show(100, 20);
}
```

Menu.removeMenuItemAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

menuInstance.removeMenuItemAt(*index*)

Paramètres

index Index de l'élément de menu à supprimer.

Valeur renvoyée

Une référence à l'élément de menu renvoyé (nœud XML). La valeur est `undefined` s'il n'y a aucun élément à cette position.

Description

Méthode : supprime l'élément de menu et tous ses enfants à l'index spécifié. Si aucun élément de menu n'existe à l'index spécifié, l'appel de cette méthode n'a aucune incidence.

Exemple

L'exemple suivant crée un menu à deux éléments, puis supprime le second élément (à l'index 1) après un intervalle de 2 secondes.

Vous devez d'abord faire glisser un composant Menu jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création d'un objet XML qui servira de jeu préconfiguré.
var my_xml:XML = new XML();

// L'élément suivant créé n'apparaît pas dans le menu.
// L'appel à la méthode createMenu() (ci-dessous) attend la
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu. Il s'agit simplement d'une méthode
// pour créer facilement cet élément racine et lui donner un nom.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Ajout des éléments de menu.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});

// Création de l'objet Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Affichage et positionnement des menus.
my_menu.show(100, 20);
var interval_id:Number = setInterval(remove, 2000, my_menu);
function remove(the_menu:Menu):Void {
  // Suppression du second élément de nœud.
  var item_obj:Object = my_menu.removeMenuItemAt(1);
  trace("Item removed: " + item_obj);
  clearInterval(interval_id);
  the_menu.show(100, 20);
}
```

Menu.rollOut

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.rollOut = function(eventObject:Object) {
  // Insertion du code ici.
};
menuInstance.addEventListener("rollOut", listenerObject);
```

Utilisation 2 :

```
on (rollOut) {
  // Insertion du code ici.
}
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque le pointeur cesse de survoler un élément de menu.

Les composants utilisent un modèle dispatcher(diffuseur)-écouteur d'événement. Lorsqu'un composant Menu diffuse un événement `rollOut`, l'événement est géré par une fonction (également appelée *gestionnaire*) associée à un objet écouteur (*`listenerObject`*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`eventObject`*) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. L'objet événement de l'événement `Menu.rollOut` possède la propriété supplémentaire suivante, `menuItem`, qui est une référence à l'élément de menu (nœud XML) que le pointeur a cessé de survoler.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

L'exemple suivant crée un menu à deux éléments et un écouteur d'événement `rollOut`. Lorsque l'événement `rollOut` est diffusé, une fonction `trace`() du gestionnaire d'événements, `menuListener`, affiche le nom de l'élément de menu pour lequel l'événement s'est produit.

Vous devez d'abord faire glisser un composant Menu jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création d'un objet XML qui servira de jeu préconfiguré.
var my_xml:XML = new XML();

// L'élément suivant créé n'apparaît pas dans le menu.
// L'appel à la méthode createMenu() (ci-dessous) attend la
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu. Il s'agit simplement d'une méthode
// pour créer facilement cet élément racine et lui donner un nom.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Ajout des éléments de menu.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});

// Création de l'objet Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Affichage et positionnement des menus.
my_menu.show(100, 20);

// Création d'un objet écouteur.
var menuListener:Object = new Object();
menuListener.rollOut = function(evt_obj:Object) {
 trace("Menu rollOut: " + evt_obj.menuItem.attributes.label);
};

// Ajout de l'écouteur.
my_menu.addEventListener("rollOut", menuListener);
```

Menu.rollOver

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.rollOver = function(eventObject:Object) {
  // Insertion du code ici.
};
menuInstance.addEventListener("rollOver", listenerObject);
```

Utilisation 2 :

```
on (rollOver) {
  // Insertion du code ici.
}
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque le pointeur survole un élément de menu.

Les composants utilisent un modèle dispatcher(diffuseur)-écouteur d'événement. Lorsqu'un composant Menu diffuse un événement `rollover`, l'événement est géré par une fonction (également appelée *gestionnaire*) associée à un objet écouteur (`listenerObject`) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (`eventObject`) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. L'objet événement de l'événement `Menu.rollOver` possède la propriété supplémentaire suivante, `menuItem`, qui est une référence à l'élément de menu (nœud XML) que le pointeur a survolé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

L'exemple suivant crée un menu à deux éléments et un écouteur d'événement `rollOver`. Lorsque l'événement `rollOver` est diffusé, une fonction `trace()` du gestionnaire d'événements, `menuListener`, affiche le nom de l'élément de menu pour lequel l'événement s'est produit.

Vous devez d'abord faire glisser un composant Menu jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création d'un objet XML qui servira de jeu préconfiguré.
var my_xml:XML = new XML();

// L'élément suivant créé n'apparaît pas dans le menu.
// L'appel à la méthode createMenu() (ci-dessous) attend la
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu. Il s'agit simplement d'une méthode
// pour créer facilement cet élément racine et lui donner un nom.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Ajout des éléments de menu.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});

// Création de l'objet Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Affichage et positionnement des menus.
my_menu.show(100, 20);

// Création d'un objet écouteur.
var menuListener:Object = new Object();
menuListener.rollOver = function(evt_obj:Object) {
 trace("Menu rollOver: "+evt_obj.menuItem.attributes.label);
};

// Ajout de l'écouteur.
my_menu.addEventListener("rollOver", menuListener);
```

Menu.setMenuItemEnabled()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

menuInstance.setMenuItemEnabled(*item*, *enable*)

Paramètres

item Nœud XML : nœud de l'élément de menu cible dans le fournisseur de données.

enable Valeur booléenne indiquant si l'élément est activé (`true`) ou non (`false`).

Valeur renvoyée

Aucune.

Description

Méthode : applique à l'attribut `enabled` de l'élément cible l'état spécifié dans le paramètre *enable*. Si cet appel entraîne un changement d'état, l'élément est redessiné avec le nouvel état.

Exemple

L'exemple suivant crée un menu à deux éléments et appelle la méthode `setMenuItemEnabled()` pour désactiver le premier.

Vous devez d'abord faire glisser un composant Menu jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création d'un objet XML qui servira de jeu préconfiguré.
var my_xml:XML = new XML();

// L'élément suivant créé n'apparaît pas dans le menu.
// L'appel à la méthode createMenu() (ci-dessous) attend la
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu. Il s'agit simplement d'une méthode
// pour créer facilement cet élément racine et lui donner un nom.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");
```

```
// Ajout des éléments de menu.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});

// Création de l'objet Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Sélection et désactivation du premier élément de menu.
var item_obj:Object = my_menu.getMenuItemAt(0);
my_menu.setMenuItemEnabled(item_obj, false);

// Affichage et positionnement du menu.
my_menu.show(100, 20);
```

Voir aussi

`Menu.setMenuItemSelected()`

Menu.setMenuItemSelected()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`menuInstance``.setMenuItemSelected(`*`item`*`, `*`select`*`)`

Paramètres

`item` Nœud XML. Le nœud de l'élément de menu cible dans le fournisseur de données.

`select` Valeur booléenne indiquant si l'élément est sélectionné (`true`) ou non (`false`). Si l'élément est une case à cocher, sa coche est visible ou invisible. Si l'élément sélectionné est un bouton radio, il devient la sélection en cours dans le groupe radio.

Valeur renvoyée

Aucune.

Description

Méthode : applique à l'attribut `selected` de l'élément l'état spécifié par le paramètre *`select`*. Si cet appel entraîne un changement d'état, l'élément est redessiné avec le nouvel état. Cette méthode n'a d'effet que sur les éléments dont l'attribut `type` est défini sur « `radio` » ou « `check` », car elle modifie leur apparence. Si vous l'appelez sur un élément de type « `normal` » ou « `separator` », aucun effet visible ne se produit.

Exemple

L'exemple suivant crée un menu avec deux éléments de menu dont le deuxième est un élément de menu check box. L'exemple appelle la méthode `setMenuItemSelected()` pour placer l'élément de menu check box dans un état sélectionné.

Vous devez d'abord faire glisser un composant Menu jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création d'un objet XML qui servira de jeu préconfiguré.
var my_xml:XML = new XML();

// L'élément suivant créé n'apparaît pas dans le menu.
// L'appel à la méthode createMenu() (ci-dessous) attend la
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu. Il s'agit simplement d'une méthode
// pour créer facilement cet élément racine et lui donner un nom.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Ajout des éléments de menu.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({type:"check", label:"2nd Item"})

// Création de l'objet Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

var myItem = my_menu.getMenuItemAt(1);
my_menu.setMenuItemSelected(myItem, true);

// Affichage et positionnement du menu.
my_menu.show(100, 20);
```

Menu.show()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
menuInstance.show(x, y)
```

Paramètres

x Coordonnée *x*.

y Coordonnée *y*.

Valeur renvoyée

Aucune.

Description

Méthode : ouvre un menu à un emplacement spécifique. Le menu est automatiquement redimensionné pour que tous les éléments de premier niveau soient visibles et que son coin supérieur gauche soit placé à l'emplacement indiqué dans le système de coordonnées fourni par le parent du composant.

Si les paramètres *x* et *y* sont omis, le menu apparaît à son emplacement précédent.

Exemple

L'exemple suivant crée un menu depuis un objet de menu XML et appelle la méthode `menu.show()` pour l'afficher.

Vous devez d'abord faire glisser un composant Menu jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

var my_xml:XML = new XML();

// Création d'éléments pour le menu.
var newItem_obj:Object = my_xml.addMenuItem({label:"New"});
my_xml.addMenuItem({label:"Open", instanceName:"miOpen"});
my_xml.addMenuItem({label:"Save", instanceName:"miSave"});
my_xml.addMenuItem({type:"separator"});
my_xml.addMenuItem({label:"Quit", instanceName:"miQuit"});

// Affichage et positionnement du menu.
var my_menu:Menu = Menu.createMenu(myParent_mc, my_xml);
my_menu.show(100, 20);
```

Voir aussi

`Menu.hide()`

Classe MenuDataProvider

Nom de classe ActionScript mx.controls.menuclasses.MenuDataProvider

La classe MenuDataProvider est une classe decorator (mix-in) qui complète la fonctionnalité de la classe globale XMLNode. Cette fonctionnalité permet aux occurrences XML affectées à une propriété `Menu.dataProvider` d'utiliser les méthodes et les propriétés MenuDataProvider pour manipuler leurs propres données et les menus associés.

REMARQUE	La classe MenuDataProvider est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Tenez compte des concepts suivants relatifs à la classe MenuDataProvider :

- MenuDataProvider est une classe decorator (mix-in). Vous ne devez pas l'instancier pour l'utiliser.
- En mode natif, les menus acceptent la valeur XML pour la propriété `dataProvider`.
- Si une classe Menu est instanciée, toutes les occurrences de XML du fichier SWF sont décorées par la classe MenuDataProvider.
- Seules les méthodes MenuDataProvider diffusent des événements aux composants Menu. Vous pouvez toutefois utiliser des méthodes XML natives, mais celles-ci ne serviront pas d'événements de diffusion destinés à actualiser l'affichage des menus. Pour contrôler le modèle de données, utilisez des méthodes MenuDataProvider. Utilisez les méthodes XML pour les opérations en lecture seule (parcourir la hiérarchie d'un menu, par exemple).
- Tous les éléments du composant Menu sont des objets XML décorés à l'aide de la classe MenuDataProvider.
- Les modifications apportées aux attributs des éléments n'apparaissent pas dans le menu tant que celui-ci n'est pas redessiné.

Méthodes de la classe MenuDataProvider

Le tableau suivant répertorie les méthodes de la classe MenuDataProvider.

Méthode	Description
`MenuDataProvider.addMenuItem()`	Ajoute un élément enfant.
`MenuDataProvider.addMenuItemAt()`	Ajoute un élément enfant à un emplacement spécifié.
`MenuDataProvider.getMenuItemAt()`	Obtient une référence à un élément de menu, à un emplacement spécifié.
`MenuDataProvider.indexOf()`	Renvoie l'index d'un élément de menu donné.
`MenuDataProvider.removeMenuItem()`	Supprime un élément de menu.
`MenuDataProvider.removeMenuItemAt()`	Supprime un élément de menu à un endroit donné.

MenuDataProvider.addMenuItem()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
myMenuDataProvider.addMenuItem(initObject)
```

Utilisation 2 :

```
myMenuDataProvider.addMenuItem(childMenuItem)
```

Paramètres

`initObject` Objet contenant les attributs qui initialisent les attributs d'un élément de menu. Pour plus d'informations, voir « Présentation des attributs XML d'élément de menu », à la page 923.

`childMenuItem` Nœud XML.

Valeur renvoyée

Une référence à un objet XMLNode.

Description

Méthode : le premier exemple d'utilisation ajoute un élément enfant à la fin d'un élément de menu parent (il peut s'agir du menu lui-même). L'élément de menu est construit à partir des valeurs transmises au paramètre *initObject*. Le second exemple d'utilisation ajoute un élément de menu enfant défini dans le paramètre XML *childMenuItem* spécifié à la fin de l'élément d'un menu parent.

Tout nœud ou élément de menu dans une occurrence MenuDataProvider peut appeler les méthodes de la classe MenuDataProvider.

Exemple

L'exemple suivant crée un menu depuis un fournisseur de données XML. Il appelle la méthode `addMenuItem()` pour ajouter deux éléments au menu principal et au sous-menu du premier élément du menu principal.

Vous devez d'abord faire glisser un composant Menu jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création d'un objet XML qui servira de jeu préconfiguré.
var my_xml:XML = new XML();

// L'élément suivant créé n'apparaît pas dans le menu.
// L'appel à la méthode createMenu() (ci-dessous) attend la
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu. Il s'agit simplement d'une méthode
// pour créer facilement cet élément racine et lui donner un nom.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");


// Ajout des éléments de menu.
menuDP_obj.addMenuItem({label:"Folders"});
menuDP_obj.addMenuItem({label:"Radio Edit", type:"radio"});

// Création de l'objet Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Affichage et positionnement du menu.
my_menu.show(100, 20);

// Récupération du premier élément de menu et ajout d'éléments à ce dernier.
var item_obj:Object = menuDP_obj.getMenuItemAt(0);
item_obj.addMenuItem({label:"First item", instanceName:"firstItem1"});
item_obj.addMenuItem({label:"Second item", instanceName:"secondItem1"});
```

MenuDataProvider.addMenuItemAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
myMenuDataProvider.addMenuItemAt(index, initObject)
```

Utilisation 2 :

```
myMenuDataProvider.addMenuItemAt(index, childMenuItem)
```

Paramètres

index Nombre entier.

objetInit Objet contenant les attributs spécifiques qui initialisent les attributs d'un élément de menu. Pour plus d'informations, voir « Présentation des attributs XML d'élément de menu », à la page 923.

childMenuItem Nœud XML.

Valeur renvoyée

Une référence au nœud XML ajouté.

Description

Méthode : le premier exemple d'utilisation ajoute un élément enfant à la position d'index indiquée dans l'élément de menu parent (il peut s'agir du menu lui-même). L'élément de menu est construit à partir des valeurs transmises au paramètre *initObject*. Le second exemple d'utilisation ajoute un élément de menu enfant défini dans le paramètre XML *childMenuItem* à l'index spécifié d'un élément de menu parent.

Tout nœud ou élément de menu dans une occurrence MenuDataProvider peut appeler les méthodes de la classe MenuDataProvider.

Exemple

L'exemple suivant crée un menu à un élément, puis appelle la méthode `addMenuItemAt()` pour ajouter un deuxième élément.

Vous devez d'abord faire glisser un composant Menu jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création d'un objet XML qui servira de jeu préconfiguré.
var my_xml:XML = new XML();

// L'élément suivant créé n'apparaît pas dans le menu.
// L'appel à la méthode createMenu() (ci-dessous) attend la
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu. Il s'agit simplement d'une méthode
// pour créer facilement cet élément racine et lui donner un nom.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Ajout des éléments de menu.
menuDP_obj.addMenuItem({label:"Edit"});

// Création de l'objet Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Affichage et positionnement du menu.
my_menu.show(100, 20);

// Ajout de l'élément de menu.
menuDP_obj.addMenuItemAt(1, {label:"Save", instanceName:"saveItem1"});
```

MenuDataProvider.getMenuItemAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myMenuDataProvider.getMenuItemAt(*index*)

Paramètres

index Nombre entier indiquant la position du menu.

Valeur renvoyée

Une référence au nœud XML spécifié.

Description

Méthode : renvoie une référence à l'élément de menu enfant spécifié dans l'élément de menu en cours.

Tout nœud ou élément de menu dans une occurrence MenuDataProvider peut appeler les méthodes de la classe MenuDataProvider.

Exemple

L'exemple suivant crée un menu, y ajoute un élément de menu, puis appelle la méthode `getMenuItemAt()` pour accéder à son objet nœud afin d'y ajouter un élément de sous-menu. Il appelle également la méthode `getMenuItemAt()` pour afficher l'étiquette de l'élément de sous-menu dans le panneau Sortie.

Vous devez d'abord faire glisser un composant Menu jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création d'un objet XML qui servira de jeu préconfiguré.
var my_xml:XML = new XML();

// L'élément suivant créé n'apparaît pas dans le menu.
// L'appel à la méthode createMenu() (ci-dessous) attend la
```

```
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu. Il s'agit simplement d'une méthode
// pour créer facilement cet élément racine et lui donner un nom.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Ajout des éléments de menu.
menuDP_obj.addMenuItem({label:"1st Item"});
var menuItem_obj:Object = menuDP_obj.getMenuItemAt(0);
menuItem_obj.addMenuItem({label:"Submenu Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});

// Création de l'objet Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Affichage et positionnement des menus.
my_menu.show(100, 20);

// Récupération de l'élément de sous-menu depuis le premier élément de menu.
var myMenuItem_obj:Object = menuDP_obj.firstChild;
trace(myMenuItem_obj.getMenuItemAt(0));
```

MenuDataProvider.indexOf()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myMenuDataProvider.indexOf(*item*)

Paramètres

item Référence au nœud XML qui décrit l'élément de menu.

Valeur renvoyée

Index de l'élément de menu spécifié. Renvoie la valeur `undefined` si l'élément ne fait pas partie de ce menu.

Description

Méthode : renvoie l'index de l'élément de menu spécifié dans cet élément de menu parent.

Tout nœud ou élément de menu dans une occurrence MenuDataProvider peut appeler les méthodes de la classe MenuDataProvider.

Exemple

L'exemple suivant ajoute un élément à un menu, puis appelle la méthode `indexOf()` pour afficher l'index de l'élément dans le panneau Sortie.

Vous devez d'abord faire glisser un composant Menu jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création d'un objet XML qui servira de jeu préconfiguré.
var my_xml:XML = new XML();

// L'élément suivant créé n'apparaît pas dans le menu.
// L'appel à la méthode createMenu() (ci-dessous) attend la
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu. Il s'agit simplement d'une méthode
// pour créer facilement cet élément racine et lui donner un nom.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Ajout des éléments de menu.
menuDP_obj.addMenuItem({label:"1st Item"});

// Création de l'objet Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Affichage et positionnement des menus.
my_menu.show(100, 20);

// Ajout d'un élément et suivi de sa position.
var myItem_obj:Object = menuDP_obj.addMenuItem({label:"That item"});
var myIndex_num:Number = menuDP_obj.indexOf(myItem_obj);
trace("Position: " + myIndex_num);
```

MenuDataProvider.removeMenuItem()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myMenuDataProvider.removeMenuItem()

Valeur renvoyée

Référence à l'élément de menu supprimé (nœud XML) ; `undefined` si une erreur se produit.

Description

Méthode : supprime l'élément cible et ses nœuds enfant, le cas échéant.

Tout nœud ou élément de menu dans une occurrence MenuDataProvider peut appeler les méthodes de la classe MenuDataProvider.

Exemple

L'exemple suivant crée un menu à trois éléments, puis appelle la méthode `removeMenuItem()` pour supprimer le premier élément de menu après un intervalle de 2 secondes (2000 millisecondes).

Vous devez d'abord faire glisser un composant Menu jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création d'un objet XML qui servira de jeu préconfiguré.
var my_xml:XML = new XML();
d
// L'élément suivant créé n'apparaît pas dans le menu.
// L'appel à la méthode createMenu() (ci-dessous) attend la
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu. Il s'agit simplement d'une méthode
// pour créer facilement cet élément racine et lui donner un nom.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Ajout des éléments de menu.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});
menuDP_obj.addMenuItem({label:"3rd Item"});

// Création de l'objet Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Affichage et positionnement des menus.
my_menu.show(100, 20);
// Appel à removeItem au bout de 2 000 millisecondes.
var interval_id:Number = setInterval(removeItem, 2000, my_menu);
function removeItem(the_menu:Menu):Void {
  // Suppression de l'élément à la position 0.
  var myItem_obj:Object = menuDP_obj.getMenuItemAt(0);
  myItem_obj.removeMenuItem();
  clearInterval(interval_id);
}
```

MenuDataProvider.removeMenuItemAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myMenuDataProvider.removeMenuItemAt(*index*)

Paramètres

index Index de l'élément de menu.

Valeur renvoyée

Une référence à l'élément de menu supprimé. La valeur est `undefined` s'il n'y a aucun élément à cette position.

Description

Méthode : supprime l'élément enfant de l'élément de menu spécifié par le paramètre *index*. Si aucun élément de menu n'existe à l'index spécifié, l'appel de cette méthode n'a aucune incidence.

Tout nœud ou élément de menu dans une occurrence MenuDataProvider peut appeler les méthodes de la classe MenuDataProvider.

Exemple

L'exemple suivant crée un menu à trois éléments, puis appelle la méthode `removeMenuItemAt()` pour supprimer le premier élément de menu après un intervalle de 2 secondes (2000 millisecondes).

Vous devez d'abord faire glisser un composant Menu jusqu'à la bibliothèque, puis ajouter le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Menu dans la bibliothèque
*/

import mx.controls.Menu;

// Création d'un objet XML qui servira de jeu préconfiguré.
var my_xml:XML = new XML();

// L'élément suivant créé n'apparaît pas dans le menu.
// L'appel à la méthode createMenu() (ci-dessous) attend la
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu. Il s'agit simplement d'une méthode
// pour créer facilement cet élément racine et lui donner un nom.
var menuDP_obj:Object = my_xml.addMenuItem("XXXXX");

// Ajout des éléments de menu.
menuDP_obj.addMenuItem({label:"1st Item"});
menuDP_obj.addMenuItem({label:"2nd Item"});
menuDP_obj.addMenuItem({label:"3rd Item"});

// Création de l'objet Menu.
var my_menu:Menu = Menu.createMenu(this, menuDP_obj);

// Affichage et positionnement des menus.
my_menu.show(100, 20);
// Appel à removeItem au bout de 2 000 millisecondes.
var interval_id:Number = setInterval(removeItem, 2000, my_menu);
function removeItem(the_menu:Menu):Void {
  // Suppression de l'élément à la position 0.
  menuDP_obj.removeMenuItemAt(0);
  clearInterval(interval_id);
}
```

CHAPITRE 31

Composant MenuBar

31

Le composant MenuBar permet de créer une barre de menus horizontale comportant des menus contextuels et des commandes, d'un type similaire à celui des barres de menu Fichier et Edition rencontrées dans les applications les plus courantes. Le composant MenuBar complète le composant Menu en fournissant une interface dans laquelle l'utilisateur peut, à l'aide de la souris ou du clavier, afficher ou masquer des menus qui se comportent comme un groupe.

REMARQUE	Le composant MenuBar est pris en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Le composant MenuBar permet de créer un menu d'application en quelques étapes seulement. Pour créer une barre de menus, vous pouvez soit affecter un fournisseur de données XML à la barre de menus pour décrire une série de menus, soit utiliser la méthode `MenuBar.addMenu()` pour ajouter des occurrences de menu de façon individuelle.

Dans la barre de menus, chaque menu se compose de deux parties : le menu et le bouton qui ouvre le menu (appelé activateur de menu). Les activateurs de menu sur lesquels vous pouvez cliquer apparaissent dans la barre de menus sous la forme d'étiquettes de texte comportant des bordures incrustées ou en relief, selon leur état de mise en valeur. Ces étiquettes réagissent aux interactions de la souris et du clavier.

Lorsque l'utilisateur clique sur l'activateur d'un menu, ce dernier s'ouvre sous l'activateur. Le menu reste actif jusqu'à ce que l'utilisateur clique à nouveau sur l'activateur, ou jusqu'à ce qu'il sélectionne un élément de menu ou clique hors de la zone du menu.

Outre les activateurs de menu, qui permettent d'afficher et de masquer les menus, le composant MenuBar regroupe plusieurs menus sous un même comportement. L'utilisateur peut ainsi parcourir un grand nombre de commandes en faisant glisser la souris sur les séries d'activateurs ou en utilisant les touches de direction pour passer d'une liste à l'autre. L'interactivité avec la souris est conjuguée à l'interactivité avec le clavier pour permettre à l'utilisateur de passer d'un menu à l'autre dans la barre de menus.

Un utilisateur ne peut pas parcourir les menus d'une barre de menus. Si des menus dépassent la largeur de la barre de menus, ils sont masqués.

Vous ne pouvez pas rendre le composant MenuBar accessible aux lecteurs d'écran.

Les menus sont souvent imbriqués dans des barres de menus. Pour plus d'informations sur les menus, reportez-vous à « Composant Menu », à la page 919.

Interaction avec le composant MenuBar

Vous pouvez utiliser la souris et le clavier pour interagir avec un composant MenuBar.

Lorsque vous survolez un activateur de menu, l'étiquette de l'activateur est mise en valeur par une bordure en relief.

Lorsqu'une occurrence de MenuBar a le focus (une fois que l'utilisateur a cliqué ou utilisé la touche de tabulation), vous pouvez utiliser les touches suivantes pour la contrôler :

Touche	Description
Flèche vers le bas	Déplace la sélection d'un élément vers le bas dans le menu.
Flèche vers le haut	Déplace la sélection d'un élément vers le haut dans le menu.
Flèche droite	Déplace la sélection vers le bouton suivant.
Flèche gauche	Déplace la sélection vers le bouton précédent.
Entrée/Echap	Ferme un menu ouvert.

REMARQUE

Lorsqu'un menu est ouvert, vous ne pouvez pas utiliser la touche de tabulation pour le fermer. Vous devez soit faire une sélection, soit fermer le menu en appuyant sur Echap.

Utilisation du composant MenuBar

Vous pouvez utiliser le composant MenuBar pour ajouter un jeu de menus (par exemple, Fichier, Edition, Spécial, Fenêtre) à la bordure supérieure d'une application.

Paramètres de MenuBar

Dans l'inspecteur des propriétés ou l'inspecteur de composants (option de menu Fenêtre > Inspecteur de composants), vous pouvez définir le paramètre de création suivant pour chaque occurrence MenuBar :

Labels Tableau qui ajoute au composant MenuBar des activateurs de menu portant les étiquettes indiquées. La valeur par défaut est `[]` (tableau vide).

Dans l'inspecteur des composants, vous pouvez définir les paramètres supplémentaires suivants pour chaque occurrence du composant MenuBar (Fenêtre > Inspecteur de composants) :

enabled Valeur booléenne qui indique si le composant peut recevoir le focus et l'entrée. La valeur par défaut est `true`.

visible Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`). La valeur par défaut est `true`.

REMARQUE

Les propriétés minHeight et minWidth sont utilisées par des routines de dimensionnement internes. Elles sont définies dans `UIObject` et remplacées par différents composants selon vos besoins. Ces propriétés peuvent être utilisées pour créer un gestionnaire de dispositions personnalisées. Dans le cas contraire, la définition de ces propriétés dans l'inspecteur de composants n'a aucun effet visible.

Vous ne pouvez pas accéder au paramètre Labels à l'aide d'ActionScript. Néanmoins, vous pouvez contrôler d'autres options du composant MenuBar à l'aide des propriétés, méthodes et événements d'ActionScript. Pour plus d'informations, voir « Classe MenuBar », à la page 991.

Création d'une application avec le composant MenuBar

Dans cet exemple, vous faites glisser un composant MenuBar sur la scène, ajoutez un code pour y ajouter des éléments de menu et associez un écouteur au menu pour répondre à la sélection d'un élément de menu.

Pour utiliser un composant MenuBar dans une application :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser le composant MenuBar du panneau Composants jusqu'à la scène.
3. Placez le menu en haut de la scène pour obtenir une disposition standard.
4. Sélectionnez l'occurrence MenuBar, puis entrez le nom d'occurrence **my_mb** dans l'inspecteur des propriétés.

5. Dans le panneau Actions, ajoutez le code suivant à l'image 1 :

```
import mx.controls.Menu;
import mx.controls.MenuBar;

var my_mb:MenuBar;

var my_menu:Menu = my_mb.addMenu("File");
my_menu.addMenuItem({label:"New", instanceName:"newInstance"});
my_menu.addMenuItem({label:"Open", instanceName:"openInstance"});
my_menu.addMenuItem({label:"Close", instanceName:"closeInstance"});
```

Ce code ajoute un menu Fichier à l'occurrence MenuBar. Il utilise ensuite une méthode Menu pour ajouter trois éléments de menu : Nouveau, Ouvrir et Fermer.

6. Dans le panneau Actions, ajoutez le code suivant à l'image 1 :

```
// Création d'un objet écouteur.
var mbListener:Object = new Object();
mbListener.change = function(evt_obj:Object) {
 var menuItem_obj:Object = evt_obj.menuItem;
 switch (menuItem_obj.attributes.instanceName) {
 case "newInstance":
  trace("New menu item");
  break;
 case "openInstance":
  trace("Open menu item");
  break;
 case "closeInstance":
  trace("Close menu item");
  break;
 }
 trace(menuItem_obj);
};

// Ajout de l'écouteur.
my_menu.addEventListener("change", mbListener);
```

Ce code crée un objet écouteur, `mblistener`, qui utilise une sélection d'élément de menu et affiche son nom et la valeur de l'objet élément de menu.

REMARQUE

Vous devez appeler la méthode `addEventListener` pour enregistrer l'écouteur avec l'occurrence de menu et non avec l'occurrence de barre de menus.

7. Choisissez Contrôle > Tester l'animation pour tester le composant MenuBar.

Personnalisation du composant MenuBar

La taille de ce composant s'ajuste automatiquement en fonction des étiquettes d'activateur fournies par la propriété `dataProvider` ou les méthodes de la classe MenuBar. Lorsqu'un bouton d'activateur se trouve sur une barre de menus, il conserve une taille fixe qui dépend des styles de police et de la longueur du texte.

Utilisation de styles avec le composant MenuBar

Le composant MenuBar crée une étiquette d'activateur pour chaque menu à l'intérieur d'un groupe. Vous pouvez utiliser des styles pour modifier l'aspect des étiquettes d'activateur. Un composant MenuBar prend en charge les styles suivants :

Style	Thème	Description
`themeColor`	Halo	Couleur de base d'un composant. `"haloGreen"`, `"haloBlue"` et `"haloOrange"` sont des valeurs possibles. La valeur par défaut est `"haloGreen"`.
`color`	Les deux	Couleur du texte. La valeur par défaut est 0x0B333C pour le thème Halo et vide pour le thème Sample.
`disabledColor`	Les deux	Couleur du texte lorsque le composant est désactivé. La couleur par défaut est 0x848384 (gris foncé).
`embedFonts`	Les deux	Valeur booléenne qui indique si la police spécifiée dans `fontFamily` est de type intégré. Ce style doit être défini sur `true` si `fontFamily` fait référence à une police intégrée. Dans le cas contraire, la police intégrée n'est pas utilisée. Si ce style est défini sur `true` et que `fontFamily` ne fait pas référence à une police intégrée, aucun texte n'est affiché. La valeur par défaut est `false`.
`fontFamily`	Les deux	Nom de la police du texte. La valeur par défaut est `"_sans"`.
`fontSize`	Les deux	Taille, en points, de la police. La valeur par défaut est 10.
`fontStyle`	Les deux	Style de police : `"normal"` ou `"italic"`. La valeur par défaut est `"normal"`.
`fontWeight`	Les deux	Epaisseur de la police : `"none"` ou `"bold"`. La valeur par défaut est `"none"`. Tous les composants peuvent également accepter la valeur `"normal"` au lieu de `"none"` pendant un appel à la méthode `setStyle()`, mais les appels suivants à `getStyle()` renvoient `"none"`.
`textDecoration`	Les deux	Décoration du texte : `"none"` ou `"underline"`. La valeur par défaut est`"none"`.

Le composant MenuBar transmet également tous les paramètres de style pour les propriétés de style Menu aux occurrences de Menu composées. Pour obtenir la liste des propriétés de style Menu, reportez-vous à « Utilisation de styles avec le composant Menu », à la page 933.

Utilisation d'enveloppes avec le composant MenuBar

Le composant MenuBar utilise trois enveloppes pour représenter son arrière-plan et un symbole de clip pour mettre les éléments individuels en surbrillance. Il contient un composant Menu servant de menu contextuel auquel une enveloppe peut être appliquée. Les enveloppes MenuBar sont décrites dans le tableau ci-dessous. Pour plus d'informations sur l'application d'enveloppe au composant Menu, reportez-vous à « Utilisation d'enveloppes avec le composant Menu », à la page 937.

Le composant MenuBar prend en charge les propriétés d'enveloppe suivantes :

Propriété	Description
`menuBarBackLeftName`	Etat relevé de l'icône contextuelle
`menuBarBackRightName`	Etat enfoncé de l'icône contextuelle
`menuBarBackMiddleName`	Etat désactivé de l'icône contextuelle

Pour créer des symboles de clip pour les enveloppes de MenuBar :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Choisissez Fichier > Importer > Ouvrir une bibliothèque externe et sélectionnez le fichier HaloTheme.fla.

   Ce fichier est stocké dans le dossier Configuration au niveau de l'application. Pour connaître son emplacement exact dans votre système d'exploitation, reportez-vous à « Présentation des thèmes » dans *Utilisation des composants ActionScript 2.0*.
3. Dans le panneau Bibliothèque du thème, choisissez Flash UI Components 2/Themes/MMDefault et faites glisser le dossier MenuBar Assets dans la bibliothèque de votre document.
4. Développez le dossier MenuBar Assets/States dans la bibliothèque de votre document.
5. Ouvrez les symboles que vous souhaitez personnaliser pour la modification.

   Par exemple, ouvrez le symbole MenuBarBackLeft.
6. Personnalisez le symbole selon vos besoins.

   Par exemple, modifiez le bord extérieur pour qu'il soit vide.

7. Répétez les étapes 5 et 6 pour tous les symboles devant être personnalisés.

   Par exemple, définissez les bords extérieurs pour les symboles du milieu et de droite de façon à ce qu'ils soient noirs.
8. Cliquez sur le bouton Précédent pour revenir au scénario principal.
9. Faites glisser un composant MenuBar sur la scène.
10. Définissez les propriétés de MenuBar de façon à ce qu'elles affichent des éléments sur la barre.
11. Sélectionnez Contrôle > Tester l'animation.

REMARQUE	La bordure utilisée pour mettre des éléments particuliers en surbrillance dans un composant MenuBar est une occurrence d'ActivatorSkin située dans le dossier Flash UI Components 2/Themes/MMDefault/Button Assets. Ce symbole peut être personnalisé pour pointer vers une classe différente afin de fournir une autre bordure. Cependant, vous ne pouvez pas modifier le nom du symbole ni utiliser un autre symbole pour des occurrences de MenuBar différentes dans un seul document.

Classe MenuBar

Héritage MovieClip > Classe UIObject > Classe UIComponent > MenuBar

Nom de classe ActionScript mx.controls.MenuBar

Les méthodes et les propriétés de la classe MenuBar vous permettent de créer une barre de menus horizontale avec des menus contextuels et des commandes. Ces méthodes et ces propriétés complètent celles de la classe Menu en vous permettant de créer une interface sur laquelle vous pouvez cliquer pour afficher et masquer, à l'aide de la souris ou du clavier, des menus qui se comportent comme un groupe.

Méthodes de la classe MenuBar

Le tableau suivant présente les méthodes de la classe MenuBar.

Méthode	Description
`MenuBar.addMenu()`	Ajoute un menu à la barre de menus.
`MenuBar.addMenuAt()`	Ajoute un menu à un endroit donné de la barre de menus.
`MenuBar.getMenuAt()`	Obtient une référence à un menu, à un emplacement donné.
`MenuBar.getMenuEnabledAt()`	Renvoie une valeur booléenne indiquant si un menu est activé (`true`) ou non (`false`).
`MenuBar.removeMenuAt()`	Supprime un menu à un emplacement donné d'une barre de menus.
`MenuBar.removeAll()`	Supprime tous les éléments de menu de la barre de menus.
`MenuBar.setMenuEnabledAt()`	Valeur booléenne indiquant si un menu peut être sélectionné (`true`) ou non (`false`).

Méthodes héritées de la classe UIObject

Le tableau suivant répertorie les méthodes de la classe MenuBar héritées de la classe UIObject. Pour appeler ces méthodes à partir de l'objet MenuBar, utilisez le formulaire `MenuBar.`*`methodName`*.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés ou des composants.
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
`UIObject.setSize()`	Redimensionne l'objet à la taille demandée.

Méthode	Description
UIObject.setSkin()	Définit une enveloppe dans l'objet.
UIObject.setStyle()	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant répertorie les méthodes de la classe MenuBar héritées de la classe UIComponent. Pour appeler ces méthodes à partir de l'objet MenuBar, utilisez le formulaire `MenuBar.methodName`.

Méthode	Description
UIComponent.getFocus()	Renvoie une référence à l'objet ayant le focus.
UIComponent.setFocus()	Attribue le focus à l'occurrence de composant.

Propriétés de la classe MenuBar

Le tableau suivant répertorie les propriétés de la classe MenuBar.

Propriété	Description
MenuBar.dataProvider	Modèle de données d'une barre de menus.
MenuBar.labelField	Chaîne déterminant l'attribut de chaque objet XMLNode à utiliser en tant que texte d'étiquette du menu.
MenuBar.labelFunction	Fonction déterminant le texte à afficher dans chaque étiquette de menu.

Propriétés héritées de la classe UIObject

Le tableau suivant répertorie les propriétés de la classe MenuBar héritées de la classe UIObject. Pour accéder à ces propriétés à partir de l'objet MenuBar, utilisez le formulaire `MenuBar.propertyName`.

Propriété	Description
UIObject.bottom	Position du bord inférieur de l'objet par rapport au bord inférieur de son parent. Lecture seule.
UIObject.height	Hauteur de l'objet, en pixels. Lecture seule.
UIObject.left	Bord gauche de l'objet, en pixels. Lecture seule.

Propriété	Description
`UIObject.right`	Position du bord droit de l'objet par rapport au bord droit de son parent. Lecture seule.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Position du bord supérieur de l'objet par rapport à son parent. Lecture seule.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Largeur de l'objet, en pixels. Lecture seule.
`UIObject.x`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.y`	Bord supérieur de l'objet, en pixels. Lecture seule.

Propriétés héritées de la classe UIComponent

Le tableau suivant répertorie les propriétés de la classe MenuBar héritées de la classe UIComponent. Pour accéder à ces propriétés à partir de l'objet MenuBar, utilisez le formulaire `MenuBar.`*`propertyName`*.

Propriété	Description
`UIComponent.enabled`	Indique si le composant peut recevoir le focus et la saisie.
`UIComponent.tabIndex`	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Evénements de la classe MenuBar

La classe MenuBar ne présente pas d'événements exclusifs.

Evénements hérités de la classe Menu

Le tableau suivant répertorie les événements de la classe MenuBar hérités de la classe Menu. Pour appeler ces événements à partir de l'objet MenuBar, utilisez le formulaire `MenuBar.eventName`.

Evénement	Description
`Menu.change`	Diffusé lorsqu'un utilisateur effectue une modification dans un menu.
`Menu.menuHide`	Diffusé lorsqu'un menu se ferme.
`Menu.menuShow`	Diffusé lorsqu'un menu s'ouvre.
`Menu.rollOut`	Diffusé lorsque le pointeur cesse de survoler un élément.
`Menu.rollOver`	Diffusé lorsque le pointeur survole un élément.

Evénements hérités de la classe UIObject

Le tableau suivant répertorie les événements de la classe MenuBar hérités de la classe UIObject. Pour appeler ces événements à partir de l'objet MenuBar, utilisez le formulaire `MenuBar.eventName`.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant répertorie les événements de la classe MenuBar hérités de la classe UIComponent. Pour appeler ces événements à partir de l'objet MenuBar, utilisez le formulaire `MenuBar.eventName`.

Evénement	Description
`UIComponent.focusIn`	Diffusé lorsqu'un objet reçoit le focus.
`UIComponent.focusOut`	Diffusé lorsqu'un objet perd le focus.
`UIComponent.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIComponent.keyUp`	Diffusé lorsqu'une touche est relâchée.

MenuBar.addMenu()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
menuBarInstance.addMenu(label)
```

Utilisation 2 :

```
menuBarInstance.addMenu(label, menuDataProvider)
```

Paramètres

label Chaîne indiquant l'étiquette du nouveau menu.

menuDataProvider Occurrence de XML ou de XMLNode décrivant le menu et ses éléments. S'il s'agit d'une occurrence de XML, le premier enfant de l'occurrence est utilisé.

Valeur renvoyée

Une référence au nouvel objet Menu.

Description

Méthode : le permier exemple d'utilisation ajoute un seul menu et un seul activateur de menu à la fin de la barre de menus et utilise l'étiquette spécifiée. Le second exemple d'utilisation ajoute un seul menu et un seul activateur de menu, définis dans le paramètre XML *menuDataProvider* spécifié.

Exemple

Utilisation 1 : L'exemple suivant ajoute un menu Fichier, puis utilise la méthode `Menu.addMenuItem()` pour ajouter les éléments de menu Nouveau et Ouvrir.

Faites glisser une occurrence du composant MenuBar sur la scène et entrez le nom d'occurrence `my_mb` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant MenuBar sur la scène (nom d'occurrence : my_mb)
*/
var my_mb:mx.controls.MenuBar;

var my_menu:mx.controls.Menu = my_mb.addMenu("File");
my_menu.addMenuItem({label:"New", instanceName:"newInstance"});
my_menu.addMenuItem({label:"Open", instanceName:"openInstance"});
```

Utilisation 2 : L'exemple suivant ajoute un menu Police contenant les éléments de menu Gras et Italique définis dans le fournisseur de données XML `myDP_xml`.

Faites glisser une occurrence du composant MenuBar sur la scène et entrez le nom d'occurrence `my_mb` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant MenuBar sur la scène (nom d'occurrence : my_mb)
*/

var my_mb:mx.controls.MenuBar;

var myDP_xml:XML = new XML();
myDP_xml.addMenuItem({type:"check", label:"Bold", instanceName:"check1"});
myDP_xml.addMenuItem({type:"check", label:"Italic",
  instanceName:"check2"});

var my_menu:mx.controls.Menu = my_mb.addMenu("Font", myDP_xml);
```

Utilisation 3 : L'exemple suivant ajoute deux menus, File (Fichier) et Edit (Edition).

Faites glisser une occurrence du composant MenuBar sur la scène et entrez le nom d'occurrence `my_mb` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant MenuBar sur la scène (nom d'occurrence : my_mb)
*/
var my_mb:mx.controls.MenuBar;

//Modification du texte de l'étiquette à lire à partir de "name".
my_mb.labelField = "name";

var my_menu:mx.controls.Menu = my_mb.addMenu({name:"File"});
my_menu.addMenuItem({name:"New", instanceName:"newInstance"});
my_menu.addMenuItem({name:"Open", instanceName:"openInstance"});

var my2_menu:mx.controls.Menu = my_mb.addMenu({name:"Edit"});
my2_menu.addMenuItem({name:"Undo", instanceName:"undoInstance"});
my2_menu.addMenuItem({name:"Redo", instanceName:"redoInstance"});
```

MenuBar.addMenuAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

`menuBarInstance.addMenuAt(`*`index`*`, `*`label`*`)`

Utilisation 2 :

`menuBarInstance.addMenuAt(`*`index`*`, `*`label`*`, `*`menuDataProvider`*`)`

Paramètres

`index` Nombre entier indiquant la position à laquelle le menu doit être inséré. La première position est 0. Pour l'ajouter à la fin de la barre de menus, appelez la méthode `MenuBar.addMenu(`*`label`*`)`.

`label` Chaîne indiquant l'étiquette du nouveau menu.

`menuDataProvider` Occurrence de XML ou de XMLNode décrivant le menu. S'il s'agit d'une occurrence de XML, le premier enfant de l'occurrence est utilisé.

Valeur renvoyée

Une référence au nouvel objet Menu.

Description

Méthode : le premier exemple d'utilisation ajoute un seul menu et un seul activateur de menu à l'index spécifié avec l'étiquette définie. Le second exemple d'utilisation ajoute un seul menu et un seul activateur de menu étiqueté à l'index spécifié. Le contenu du menu est défini dans le paramètre *menuDataProvider*.

Exemple

Utilisation 1 : L'exemple suivant place un menu en première position de l'occurrence MenuBar `my_mb`.

Faites glisser une occurrence du composant MenuBar sur la scène et entrez le nom d'occurrence `my_mb` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant MenuBar sur la scène (nom d'occurrence : my_mb)
*/

var my_mb:mx.controls.MenuBar;

var my_menu:mx.controls.Menu = my_mb.addMenuAt(0, "Flash");
my_menu.addMenuItem({label:"About Adobe Flash", instanceName:"aboutInst"});
my_menu.addMenuItem({label:"Preferences", instanceName:"PrefInst"});
```

Utilisation 2 : L'exemple suivant ajoute un menu Edition comportant les éléments de menu Annuler, Répéter, Couper et Copier, définis dans le fournisseur de données XML `myDP_xml`. Il ajoute le menu à la première position de l'occurrence MenuBar `my_mb`.

Faites glisser une occurrence du composant MenuBar sur la scène et entrez le nom d'occurrence `my_mb` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant MenuBar sur la scène (nom d'occurrence : my_mb)
*/

var my_mb:mx.controls.MenuBar;

var myDP_xml:XML = new XML();
myDP_xml.addMenuItem({label:"Undo", instanceName:"undoInst"});
myDP_xml.addMenuItem({label:"Redo", instanceName:"redoInst"});
myDP_xml.addMenuItem({type:"separator"});
myDP_xml.addMenuItem({label:"Cut", instanceName:"cutInst"});
myDP_xml.addMenuItem({label:"Copy", instanceName:"copyInst"});

my_mb.addMenuAt(0, "Edit", myDP_xml);
```

MenuBar.dataProvider

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
menuBarInstance.dataProvider
```

Description

Propriété : le modèle de données des éléments d'un composant MenuBar.

`MenuBar.dataProvider` est un objet nœud XML. La définition de cette propriété remplace le modèle de données actuel du composant MenuBar. Si le fournisseur de données possède des nœuds enfants, ceux-ci deviennent les éléments de la barre de menus ; les sous-nœuds de ces nœuds enfant deviennent les éléments de leurs menus respectifs.

La valeur par défaut est `undefined`.

REMARQUE	Toutes les occurrences de XML ou de XMLNode héritent automatiquement des méthodes et des propriétés de la classe MenuDataProvider lorsqu'elles sont utilisées avec le composant MenuBar.

Exemple

L'exemple suivant charge un fichier de menu XML à partir d'une page Web et utilise le gestionnaire d'événements `onLoad` pour l'affecter à la propriété `dataProvider` de l'occurrence MenuBar `my_mb`.

Faites glisser une occurrence du composant MenuBar sur la scène et entrez le nom d'occurrence `my_mb` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant MenuBar sur la scène (nom d'occurrence : my_mb)
*/

var my_mb:mx.controls.MenuBar;

var myDP_xml:XML = new XML();
myDP_xml.ignoreWhite = true;
```

```
myDP_xml.onLoad = function(success:Boolean) {
 if (success) {
  my_mb.dataProvider = myDP_xml.firstChild;
 } else {
  trace("error loading XML file");
 }
};
myDP_xml.load("http://www.flash-mx.com/mm/xml/menubar.xml");
```

MenuBar.getMenuAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`menuBarInstance.getMenuAt(`*`index`*`)`

Paramètres

`index` Nombre entier indiquant la position du menu.

Valeur renvoyée

Une référence au menu à l'index spécifié. Cette valeur est `undefined` s'il n'y a aucun menu à cette position.

Description

Méthode : renvoie une référence au menu à l'index spécifié. Etant donné que la méthode `getMenuAt()` renvoie une référence, vous pouvez ajouter des éléments dans un menu à l'index spécifié.

Exemple

L'exemple suivant créé un menu Fichier et appelle la méthode `getMenuAt()` pour lui créer une référence. Il utilise ensuite la référence pour ajouter deux éléments de menu, Nouveau et Ouvrir, au menu Fichier.

Faites glisser une occurrence du composant MenuBar sur la scène et entrez le nom d'occurrence `my_mb` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant MenuBar sur la scène (nom d'occurrence : my_mb)
*/

var my_mb:mx.controls.MenuBar;

my_mb.addMenu("File");

var my_menu:mx.controls.Menu = my_mb.getMenuAt(0);
my_menu.addMenuItem({label:"New",instanceName:"newInst"});
my_menu.addMenuItem({label:"Open",instanceName:"openInst"});
```

MenuBar.getMenuEnabledAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
menuBarInstance.getMenuEnabledAt(index)
```

Paramètres

index Index du menu de la barre de menus.

Valeur renvoyée

Une valeur booléenne indiquant si le menu peut être choisi (`true`) ou non (`false`).

Description

Méthode : renvoie une valeur booléenne qui indique si le menu peut être choisi (`true`) ou non (`false`).

Exemple

L'exemple suivant crée un menu Fichier comportant deux éléments, puis appelle la méthode `setMenuEnabledAt()` avec la valeur `false` pour le désactiver. Il appelle également la méthode `getMenuEnabledAt()` et affiche le résultat pour indiquer comment déterminer si un menu est activé ou non.

Faites glisser une occurrence du composant MenuBar sur la scène et entrez le nom d'occurrence `my_mb` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant MenuBar sur la scène (nom d'occurrence : my_mb)
*/

var my_mb:mx.controls.MenuBar;

var my_menu:mx.controls.Menu = my_mb.addMenu("File");
my_menu.addMenuItem({label:"New", instanceName:"newInstance"});
my_menu.addMenuItem({label:"Open", instanceName:"openInstance"});

//Désactivation du menu « fichier ».
my_mb.setMenuEnabledAt(0, false);

//Vérification du fait que le menu « fichier » peut être sélectionné ou non.
trace("Menu can be selected: " + my_mb.getMenuEnabledAt(0));
```

MenuBar.labelField

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
menuBarInstance.labelField
```

Description

Propriété ; chaîne déterminant l'attribut de chaque XMLNode à utiliser en tant que texte d'étiquette du menu. La valeur de cette propriété est également transmise à tous les menus créés à partir de la barre de menus. La valeur par défaut est « `label` ».

Une fois que la propriété `dataProvider` est définie, elle est en lecture seule.

Exemple

L'exemple suivant indique que l'attribut `name` de chaque nœud XML doit fournir le texte de l'étiquette pour les éléments de menu.

Faites glisser une occurrence du composant MenuBar sur la scène et entrez le nom d'occurrence `my_mb` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant MenuBar sur la scène (nom d'occurrence : my_mb)
*/
var my_mb:mx.controls.MenuBar;

//Modification du texte de l'étiquette à lire de « name ».
my_mb.labelField = "name";

var my_menu:mx.controls.Menu = my_mb.addMenu({name:"File"});
my_menu.addMenuItem({name:"New", instanceName:"newInstance"});
my_menu.addMenuItem({name:"Open", instanceName:"openInstance"});
```

MenuBar.labelFunction

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
menuBarInstance.labelFunction
```

Description

Propriété ; une fonction déterminant le texte à afficher dans chaque étiquette de menu. Cette fonction accepte le nœud XML associé à l'élément en tant que paramètre et renvoie une chaîne à utiliser en tant que texte d'étiquette. Cette propriété est transmise à tous les menus créés à partir de la barre de menus. La valeur par défaut est `undefined`.

Une fois que la propriété `dataProvider` est définie, elle est en lecture seule.

Exemple

L'exemple suivant utilise une fonction label pour créer et renvoyer une étiquette personnalisée (par exemple, Nouveau (Ctrl +N), depuis les attributs de nœud.

Faites glisser une occurrence du composant MenuBar sur la scène et entrez le nom d'occurrence `my_mb` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant MenuBar sur la scène (nom d'occurrence : my_mb)
*/

var my_mb:mx.controls.MenuBar;

var my_menu:mx.controls.Menu = my_mb.addMenu("File");
my_menu.addMenuItem({label:"New", data:"Control+N",
  instanceName:"newInstance"});
my_menu.addMenuItem({label:"Open", data:"Control+O",
  instanceName:"openInstance"});
my_menu.addMenuItem({label:"Close", data:"Control+W",
  instanceName:"closeInstance"});

//Formatage des données XML fournies pour le menu.
my_menu.labelFunction = function(node:XMLNode):String  {
 var attrb:Object = node.attributes;
 return (attrb.label + " (" + attrb.data + ")");
};
```

MenuBar.removeAll()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
menuBarInstance.removeAll()
```

Paramètres

Aucun.

Valeur renvoyée

Aucune.

Description

Méthode ; supprime tous les éléments de menu de la barre de menus.

Exemple

L'exemple suivant crée les menus Fichier, Edition, Outils et Fenêtre sur la barre de menus. Ensuite, lorsque vous cliquez sur un bouton, le script appelle la méthode `removeAll()` pour supprimer les éléments de menu.

Faites glisser le composant MenuBar sur la scène et entrez le nom d'occurrence `myMenuBar` dans l'inspecteur des propriétés. Faites également glisser le composant Button sur la scène et entrez le nom d'occurrence `remBtn`. Ajoutez le code suivant à l'image 1 du scénario :

```
var menu = myMenuBar.addMenu("File");
var menu = myMenuBar.addMenu("Edit");
var menu = myMenuBar.addMenu("Tools");
var menu = myMenuBar.addMenu("Window");
// Ajout d'un bouton qui supprime les éléments de menu.
var rem_listener = new Object();
rem_listener.click = function() {
  myMenuBar.removeAll();
};
remBtn.addEventListener("click", rem_listener);
```

MenuBar.removeMenuAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`menuBarInstance.removeMenuAt(`*`index`*`)`

Paramètres

`index` Index du menu à supprimer de la barre de menus.

Valeur renvoyée

Une référence au menu à l'index spécifié dans la barre de menus. La valeur est `undefined` s'il n'y a aucun menu à cette position dans la barre de menus.

Description

Méthode : supprime l'élément de menu à l'index spécifié. Si aucun élément de menu n'existe à l'index spécifié, l'appel de cette méthode n'a aucune incidence. De plus, lorsque vous supprimez plusieurs menus, les affectations d'index sont décalées en conséquence lorsque vous supprimez chaque menu.

Exemple

L'exemple suivant crée un menu File (Fichier) et un menu Edit (Edition) dans la barre de menus. Il appelle ensuite la méthode `removeMenuAt()` pour supprimer le menu à la position 0 qui se trouve être le menu File, et laisse le menu Edit.

Faites glisser une occurrence du composant MenuBar sur la scène et entrez le nom d'occurrence `my_mb` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant MenuBar sur la scène (nom d'occurrence : my_mb)
*/

import mx.controls.Menu;
import mx.controls.MenuBar;

var my_mb:MenuBar;

var file_menu:Menu = my_mb.addMenu("File");
file_menu.addMenuItem({label:"New", instanceName:"newInstance"});
file_menu.addMenuItem({label:"Open", instanceName:"openInstance"});

var edit_menu:Menu = my_mb.addMenu("Edit");
edit_menu.addMenuItem({label:"Cut", instanceName:"cutInstance"});
edit_menu.addMenuItem({label:"Copy", instanceName:"copyInstance"});
edit_menu.addMenuItem({label:"Paste", instanceName:"pasteInstance"});

//Suppression du menu « fichier ».
my_mb.removeMenuAt(0);
```

MenuBar.setMenuEnabledAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`menuBarInstance.setMenuEnabledAt(`*`index`*`, `*`boolean`*`)`

Paramètres

`index` Index de l'élément de menu à définir dans l'occurrence de MenuBar.

`boolean` Valeur booléenne indiquant si l'élément de menu à l'index spécifié est activé (`true`) ou non (`false`).

Valeur renvoyée

Aucune.

Description

Méthode : active le menu à l'index spécifié. Si aucun menu n'existe à cet index, l'appel de cette méthode n'a aucune incidence.

Exemple

L'exemple suivant ajoute un menu Fichier à la barre de menus et appelle la méthode `setMenuEnabledAt()` pour activer ou désactiver le menu, selon que la case à cocher menuEnabled_ch est activée ou non.

Faites glisser une occurrence du composant MenuBar sur la scène et entrez le nom d'occurrence **my_mb** dans l'inspecteur des propriétés. Faites glisser un composant CheckBox sur la scène et nommez l'occurrence **menuEnabled_ch**. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant MenuBar sur la scène (nom d'occurrence : my_mb)
  - composant CheckBox sur la scène (nom d'occurrence : menuEnabled_ch)
*/

import mx.controls.CheckBox;
import mx.controls.Menu;
import mx.controls.MenuBar;

var my_mb:MenuBar;
var menuEnabled_ch:CheckBox;

menuEnabled_ch.selected = true;
var my_menu:Menu = my_mb.addMenu("File");
my_menu.addMenuItem({label:"New", instanceName:"newInstance"});
my_menu.addMenuItem({label:"Open", instanceName:"openInstance"});

var chListener:Object = new Object();
chListener.click = function(evt_obj:Object) {
 // Basculement du menu « fichier ».
my_mb.setMenuEnabledAt(0, evt_obj.target.selected);
}
menuEnabled_ch.addEventListener("click", chListener);
```

CHAPITRE 32

Composant NumericStepper 32

Le composant NumericStepper permet à un utilisateur de faire défiler un ensemble ordonné de nombres. Il s'agit d'un nombre dans une zone de texte affiché à côté de petits boutons fléchés vers le haut et vers le bas. Lorsqu'un utilisateur appuie sur les boutons, le nombre augmente ou diminue de façon incrémentielle en fonction de l'unité spécifiée dans le paramètre `stepSize` jusqu'à ce que l'utilisateur relâche les boutons ou que la valeur maximale ou minimale soit atteinte. Le texte dans la zone de texte du composant NumericStepper peut également être modifié.

REMARQUE

Un composant NumericStepper est pris en charge pour ActionScript 2.0 et ActionScript 3.0. Ce document présente le composant de la version 2. Si vous utilisez celui de la version 3, reportez-vous à « Utilisation du composant NumericStepper » dans *Utilisation des composants ActionScript 3.0*.

Le composant NumericStepper gère uniquement les données numériques. Vous devez également redimensionner l'incrémenteur lors de la programmation pour afficher plus de deux chiffres (par exemple, les nombres 5246 ou 1,34).

Un incrémenteur peut être activé ou désactivé dans une application. Lorsqu'il est désactivé, le stepper ne reçoit aucune information provenant du clavier ou de la souris. Un incrémenteur activé reçoit le focus si vous cliquez dessus ou si vous utilisez la tabulation pour l'ouvrir et son focus interne est défini dans la zone de texte. Lorsqu'une occurrence de NumericStepper a le focus, vous pouvez utiliser les touches suivantes pour la contrôler :

Touche	Description
Flèche vers le bas	La valeur est modifiée d'une unité.
Flèche gauche	Déplace le point d'insertion vers la gauche à l'intérieur de la zone de texte.
Flèche droite	Déplace le point d'insertion vers la droite à l'intérieur de la zone de texte.
Maj+Tab	Place le focus sur l'objet précédent.

Touche	Description
Tab	Place le focus sur l'objet suivant.
Flèche vers le haut	La valeur est modifiée d'une unité.

Pour plus d'informations sur le contrôle du focus, reportez-vous à « Classe FocusManager », à la page 745 ou à « Création de la navigation personnalisée du focus » dans *Utilisation des composants ActionScript 2.0*.

Un aperçu en direct de chaque occurrence d'incrémenteur reflète la valeur du paramètre value dans l'inspecteur des propriétés ou des composants pendant la programmation. Cependant, il n'y a aucune interaction entre le clavier ou la souris et les boutons fléchés de l'incrémenteur dans l'aperçu en direct.

Lorsque vous ajoutez le composant NumericStepper à une application, vous pouvez utiliser le panneau Accessibilité pour le rendre accessible aux lecteurs d'écran. Vous devez d'abord ajouter la ligne de code suivante pour activer l'accessibilité :

```
mx.accessibility.NumericStepperAccImpl.enableAccessibility();
```

Quel que soit le nombre d'occurrences d'un composant, l'activation de son accessibilité ne se fait qu'une fois. Pour plus d'informations, reportez-vous à *Création de contenu accessible* dans *Utilisation de Flash*.

Utilisation du composant NumericStepper

Vous pouvez utiliser le NumericStepper là où vous souhaitez qu'un utilisateur sélectionne une valeur numérique. Par exemple, vous pouvez utiliser un composant NumericStepper dans un formulaire pour permettre à un utilisateur de définir la date d'expiration d'une carte bancaire. Vous pouvez également utiliser un composant NumericStepper pour permettre à un utilisateur d'augmenter ou de diminuer la taille d'une police.

Paramètres de NumericStepper

Dans l'inspecteur des propriétés ou l'inspecteur de composants (Fenêtre > Inspecteur de composants), vous pouvez définir les paramètres de création suivants pour chaque occurrence NumericStepper :

maximum définit la valeur maximale pouvant être affichée dans l'incrémenteur. La valeur par défaut est définie sur 10. Si vous définissez une unité stepSize afin que la valeur minimale plus la valeur stepSize à ce point n'excèdent pas la valeur maximale (minimum + stepSize + stepSize + stepSize, etc.), la valeur maximale *s'affichera* lors du dépassement du maximum par l'incrémenteur.

minimum définit la valeur minimale pouvant être affichée dans l'incrémenteur. La valeur par défaut est 0.

stepSize définit de combien l'incrémenteur augmente ou diminue à chaque clic. La valeur par défaut est 1.

value définit la valeur affichée dans la zone de texte de l'incrémenteur. La valeur par défaut est 0.

Dans l'inspecteur des composants, vous pouvez définir les paramètres supplémentaires suivants pour chaque occurrence du composant NumericStepper (Fenêtre > Inspecteur de composants) :

enabled est une valeur booléenne qui indique si le composant peut recevoir le focus et l'entrée. La valeur par défaut est `true`.

visible est une valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`). La valeur par défaut est `true`.

REMARQUE	Les propriétés minHeight et minWidth sont utilisées par des routines de dimensionnement internes. Elles sont définies dans UIObject et remplacées par différents composants selon vos besoins. Ces propriétés peuvent être utilisées pour créer un gestionnaire de dispositions personnalisées. Dans le cas contraire, la définition de ces propriétés dans l'inspecteur de composants n'a aucun effet visible.

Vous pouvez rédiger du code ActionScript pour contrôler ces options et d'autres options du composant NumericStepper à l'aide des propriétés, méthodes et événements ActionScript. Pour plus d'informations, voir « Classe NumericStepper », à la page 1017.

Création d'une application avec le composant NumericStepper

La procédure suivante explique comment ajouter un composant NumericStepper à une application lors de la programmation. L'exemple place un composant NumericStepper et un composant Label sur la scène et crée un écouteur pour un événement change sur l'occurrence de composant NumericStepper. Lorsque la valeur dans l'incrémenteur numérique change, l'exemple affiche la nouvelle valeur dans l'occurrence de composant Label.

Pour créer une application avec le composant NumericStepper :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser un composant NumericStepper du panneau Composants jusqu'à la scène.
3. Dans l'inspecteur Propriétés, saisissez le nom d'occurrence **my_nstep**.
4. Faites glisser un composant Label du panneau Composants vers la scène.
5. Dans l'inspecteur Propriétés, saisissez le nom d'occurrence **my_label**.
6. Sélectionnez l'image 1 dans le scénario, ouvrez le panneau Actions et saisissez le code suivant :

```
/**
 Requiert :
  - Composant NumericStepper sur la scène (nom d'occurrence : my_nstep)
  - composant Label sur la scène (nom d'occurrence : my_label)
*/

var my_nstep:mx.controls.NumericStepper;
var my_label:mx.controls.Label;

my_label.text = "value = " + my_nstep.value;

// Création d'un objet écouteur.
var nstepListener:Object = new Object();
nstepListener.change = function(evt_obj:Object) {
 my_label.text = "value = " + evt_obj.target.value;
};

// Ajout de l'écouteur.
my_nstep.addEventListener("change", nstepListener);
```

   La dernière ligne de code ajoute un gestionnaire d'événements `change` à l'occurrence `my_nstep`. Le gestionnaire (`nstepListener`) affecte la valeur actuelle dans l'incrémenteur numérique à la propriété text de l'occurrence Label.
7. Sélectionnez Contrôle > Tester l'animation.

Personnalisation du composant NumericStepper

Vous pouvez orienter un composant NumericStepper horizontalement et verticalement pendant la programmation et lors de l'exécution. Lors de la programmation, sélectionnez le composant sur la scène et utilisez l'outil Transformer librement ou une commande Modification > Transformer. Lors de l'exécution, utilisez la méthode `setSize()` (reportez-vous à `UIObject.setSize()`) ou à toutes les propriétés et méthodes applicables de la classe NumericStepper. (Voir la section « Classe NumericStepper », à la page 1017).

Le redimensionnement du composant NumericStepper ne modifie pas la taille des boutons fléchés vers le haut et le bas. Si l'incrémenteur est redimensionné au-delà de la hauteur par défaut, les boutons fléchés sont placés en haut et en bas du composant. Les boutons fléchés apparaissent toujours à droite de la zone de texte.

Utilisation de styles avec le composant NumericStepper

Vous pouvez définir des propriétés de style afin de modifier l'aspect d'une occurrence de NumericStepper. Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, consultez « Utilisation de styles pour personnaliser la couleur et le texte des composants » dans *Utilisation des composants ActionScript 2.0*.

Un composant NumericStepper prend en charge les styles suivants :

Style	Thème	Description
`themeColor`	Halo	Couleur de base d'un composant. `"haloGreen"`, `"haloBlue"` et `"haloOrange"` sont des valeurs possibles. La valeur par défaut est `"haloGreen"`.
`color`	Les deux	Couleur du texte. La valeur par défaut est 0x0B333C pour le thème Halo et vide pour le thème Sample.
`disabledColor`	Les deux	Couleur du texte lorsque le composant est désactivé. La couleur par défaut est 0x848384 (gris foncé).

Style	Thème	Description
`embedFonts`	Les deux	Valeur booléenne qui indique si la police spécifiée dans `fontFamily` est de type intégré. Ce style doit être défini sur `true` si `fontFamily` fait référence à une police intégrée. Dans le cas contraire, la police intégrée n'est pas utilisée. Si ce style est défini sur `true` et que `fontFamily` ne fait pas référence à une police intégrée, aucun texte n'est affiché. La valeur par défaut est `false`.
`fontFamily`	Les deux	Nom de la police du texte. La valeur par défaut est `"_sans"`.
`fontSize`	Les deux	Taille, en points, de la police. La valeur par défaut est 10.
`fontStyle`	Les deux	Style de police : `"normal"` ou `"italic"`. La valeur par défaut est `"normal"`.
`fontWeight`	Les deux	Epaisseur de la police : `"none"` ou `"bold"`. La valeur par défaut est `"none"`. Tous les composants peuvent également accepter la valeur « `normal` » au lieu de « `none` » pendant un appel de `setStyle()`, mais les prochains appels de `getStyle()` renvoient « `none` ».
`textAlign`	Les deux	Alignement du texte : « `left` », « `right` » ou « `center` ». La valeur par défaut est `"center"`.
`textDecoration`	Les deux	Décoration du texte : `"none"` ou `"underline"`. La valeur par défaut est `"none"`.
`repeatDelay`	Les deux	Nombre de millisecondes de délai entre le moment où un utilisateur appuie sur un bouton pour la première fois et le moment où l'action commence à se répéter. La valeur par défaut est 500 (une demi-seconde).
`repeatInterval`	Les deux	Nombre de millisecondes entre les clics automatiques lorsqu'un utilisateur maintient le bouton de la souris enfoncé sur un bouton. La valeur par défaut est 35.
`symbolColor`	Sample	Couleur des flèches. La valeur par défaut est 0x2B333C (gris foncé).

Utilisation d'enveloppes avec le composant NumericStepper

Le composant NumericStepper utilise des enveloppes pour représenter les états relevé et enfoncé de ses boutons. Pour appliquer une enveloppe à un composant NumericStepper lors de la programmation, modifiez les symboles d'enveloppes dans le dossier Flash UI Components 2/Themes/MMDefault/Stepper Assets/States de la bibliothèque. Pour plus d'informations, consultez « A propos de l'application d'enveloppes aux composants » dans *Utilisation des composants ActionScript 2.0*.

Si un incrémenteur est activé, les boutons fléchés vers le haut et vers le bas affichent leur état survolé lorsque le pointeur se déplace au-dessus d'eux. Les boutons affichent leur état enfoncé lorsque l'utilisateur clique dessus. Les boutons reviennent à l'état survolé lorsque le bouton de la souris est relâché. Si le pointeur s'éloigne des boutons alors que le bouton de la souris est enfoncé, les boutons reviennent à leur état original.

Si un incrémenteur est désactivé, il affiche son état désactivé, quelle que soit l'interaction de l'utilisateur.

Un composant NumericStepper prend en charge les propriétés d'enveloppe suivantes :

Propriété	Description
`upArrowUp`	L'état haut du bouton fléché vers le haut. La valeur par défaut est `StepUpArrowUp`.
`upArrowDown`	Etat enfoncé du bouton fléché vers le haut. La valeur par défaut est `StepUpArrowDown`.
`upArrowOver`	Etat Survolé du bouton fléché vers le haut. La valeur par défaut est `StepUpArrowOver`.
`upArrowDisabled`	Etat désactivé du bouton fléché vers le haut. La valeur par défaut est `StepUpArrowDisabled`.
`downArrowUp`	Etat Relevé du bouton fléché vers le bas. La valeur par défaut est `StepDownArrowUp`.
`downArrowDown`	Etat Enfoncé du bouton fléché vers le bas. La valeur par défaut est `StepDownArrowDown`.
`downArrowOver`	Etat Survolé du bouton fléché vers le bas. La valeur par défaut est `StepDownArrowOver`.
`downArrowDisabled`	Etat désactivé du bouton fléché vers le bas. La valeur par défaut est `StepDownArrowDisabled`.

Pour créer des symboles de clip pour les enveloppes NumericStepper :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Choisissez Fichier > Importer > Ouvrir une bibliothèque externe et sélectionnez le fichier HaloTheme.fla.

   Ce fichier est stocké dans le dossier Configuration au niveau de l'application. Pour connaître son emplacement exact dans votre système d'exploitation, reportez-vous à « Présentation des thèmes » dans *Utilisation des composants ActionScript 2.0*.
3. Dans le panneau Bibliothèque du thème, choisissez Flash UI Components 2/Themes/MMDefault et faites glisser le dossier Stepper Assets dans la bibliothèque de votre document.
4. Développez le dossier Stepper Assets dans la bibliothèque de votre document.
5. Développez le dossier Stepper Assets/States dans la bibliothèque de votre document.
6. Ouvrez les symboles que vous souhaitez personnaliser pour la modification.

   Par exemple, ouvrez le symbole StepDownArrowDisabled.
7. Personnalisez le symbole selon vos besoins.

   Par exemple, appliquez la couleur gris clair aux graphiques blancs intérieurs.
8. Répétez les étapes 6 et 7 pour tous les symboles devant être personnalisés.

   Par exemple, appliquez le même changement à la flèche vers le haut.
9. Cliquez sur le bouton Précédent pour revenir au scénario principal.
10. Faites glisser un composant NumericStepper sur la scène.

    Cet exemple a personnalisé les enveloppes désactivées. Par conséquent, utilisez ActionScript pour définir l'occurrence de NumericStepper à désactiver de façon à visualiser les enveloppes modifiées.
11. Sélectionnez Contrôle > Tester l'animation.

REMARQUE	Le dossier Stepper Assets/States contient également un symbole StepTrack utilisé comme séparateur entre les enveloppes supérieure et inférieure si la hauteur totale de l'occurrence NumericStepper est supérieure à la somme des deux hauteurs fléchées. Cet identificateur de liaison de symbole ne peut pas être modifié à l'aide d'une propriété d'enveloppe mais le symbole de bibliothèque peut être modifié tant que l'identificateur de liaison demeure inchangé.

Classe NumericStepper

Héritage MovieClip > Classe UIObject > Classe UIComponent > NumericStepper

Nom de classe ActionScript mx.controls.NumericStepper

Les propriétés de la classe NumericStepper permettent de définir les éléments suivants lors de l'exécution : les valeurs minimum et maximum affichées dans l'incrémenteur, de combien ce dernier augmente ou diminue en réponse à un clic et la valeur courante qu'il affiche.

La définition d'une propriété de la classe NumericStepper avec ActionScript annule le paramètre du même nom défini dans l'inspecteur des propriétés ou des composants.

Le composant NumericStepper utilise le gestionnaire de focus pour remplacer le rectangle de focus par défaut de Flash Player et tracer un rectangle de focus personnalisé aux coins arrondis. Pour plus d'informations, consultez « Création de la navigation personnalisée du focus » dans *Utilisation des composants ActionScript 2.0*.

Toutes les classes de composants ont une propriété `version` qui correspond à une propriété de classe. Les propriétés de classe ne sont disponibles que pour la classe elle-même. La propriété `version` renvoie une chaîne qui indique la version du composant. Pour accéder à cette propriété, utilisez le code suivant :

```
trace(mx.controls.NumericStepper.version);
```

REMARQUE

Le code `trace(myNumericStepperInstance.version);` renvoie `undefined`.

Méthodes de la classe NumericStepper

La classe NumericStepper ne présente pas de méthodes exclusives.

Méthodes héritées de la classe UIObject

Le tableau suivant répertorie les méthodes de la classe NumericStepper héritées de la classe UIObject. Pour appeler ces méthodes à partir de l'objet NumericStepper, utilisez le formulaire `NumericStepper.methodName`.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
`UIObject.setSize()`	Redimensionne l'objet à la taille demandée.
`UIObject.setSkin()`	Définit une enveloppe dans l'objet.
`UIObject.setStyle()`	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant répertorie les méthodes de la classe NumericStepper héritées de la classe UIComponent. Pour appeler ces méthodes à partir de l'objet NumericStepper, utilisez le formulaire `NumericStepper.methodName`.

Méthode	Description
`UIComponent.getFocus()`	Renvoie une référence à l'objet ayant le focus.
`UIComponent.setFocus()`	Attribue le focus à l'occurrence de composant.

Propriétés de la classe NumericStepper

Le tableau suivant répertorie les propriétés de la classe NumericStepper.

Propriété	Description
`NumericStepper.maximum`	Nombre indiquant la valeur de plage maximum.
`NumericStepper.minimum`	Nombre indiquant la valeur de plage minimum.
`NumericStepper.nextValue`	Nombre indiquant la prochaine valeur séquentielle. Cette propriété est en lecture seule.
`NumericStepper.previousValue`	Nombre indiquant la valeur séquentielle précédente. Cette propriété est en lecture seule.
`NumericStepper.stepSize`	Nombre indiquant l'unité de changement pour chaque clic.
`NumericStepper.value`	Nombre indiquant la valeur courante de l'incrémenteur.

Propriétés héritées de la classe UIObject

Le tableau suivant répertorie les propriétés de la classe NumericStepper héritées de la classe UIObject. Pour appeler ces propriétés à partir de l'objet NumericStepper, utilisez le formulaire `NumericStepper.`*`propertyName`*.

Propriété	Description
`UIObject.bottom`	Position du bord inférieur de l'objet par rapport au bord inférieur de son parent. Lecture seule.
`UIObject.height`	Hauteur de l'objet, en pixels. Lecture seule.
`UIObject.left`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.right`	Position du bord droit de l'objet par rapport au bord droit de son parent. Lecture seule.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Position du bord supérieur de l'objet par rapport à son parent. Lecture seule.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Largeur de l'objet, en pixels. Lecture seule.
`UIObject.x`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.y`	Bord supérieur de l'objet, en pixels. Lecture seule.

Propriétés héritées de la classe UIComponent

Le tableau suivant répertorie les propriétés de la classe NumericStepper héritées de la classe UIComponent. Pour appeler ces propriétés à partir de l'objet NumericStepper, utilisez le formulaire `NumericStepper.propertyName`.

Propriété	Description
`UIComponent.enabled`	Indique si le composant peut recevoir le focus et la saisie.
`UIComponent.tabIndex`	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Evénements de la classe NumericStepper

Le tableau suivant présente l'événement de la classe NumericStepper.

Evénement	Description
`NumericStepper.change`	Déclenché lorsque la valeur de l'incrémenteur change.

Evénements hérités de la classe UIObject

Le tableau suivant répertorie les événements de la classe NumericStepper hérités de la classe UIObject. Pour appeler ces événements à partir de l'objet NumericStepper, utilisez le formulaire `NumericStepper.eventName`.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant répertorie les événements de la classe NumericStepper hérités de la classe UIComponent. Pour appeler ces événements à partir de l'objet NumericStepper, utilisez le formulaire `NumericStepper.eventName`.

Evénement	Description
`UIComponent.focusIn`	Diffusé lorsqu'un objet reçoit le focus.
`UIComponent.focusOut`	Diffusé lorsqu'un objet perd le focus.
`UIComponent.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIComponent.keyUp`	Diffusé lorsqu'une touche est relâchée.

NumericStepper.change

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.change = function(eventObject:Object) {
  //...
};
numericStepperInstance.addEventListener("change", listenerObject);
```

Utilisation 2 :

```
on (change) {
  // ...
}
```

Description

Evénement : diffusé à l'ensemble des écouteurs enregistrés lorsque la valeur de l'incrémenteur est modifiée.

Une occurrence de composant (*stepperInstance*) distribue un événement (ici, change) qui est géré par une fonction (également appelée *gestionnaire*), sur un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée au déclenchement de l'événement. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. Chaque objet événement est doté de propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

L'exemple suivant crée un écouteur pour un événement `change` sur l'incrémenteur numérique appelé `my_nstep`. Lorsque vous modifiez la valeur dans l'incrémenteur numérique, l'écouteur affiche la valeur (propriété `value`) dans le panneau Sortie.

Faites glisser une occurrence du composant NumericStepper sur la scène et nommez l'occurrence `my_nstep` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant NumericStepper sur la scène (nom d'occurrence : my_nstep)
*/

var my_nstep:mx.controls.NumericStepper;

// Création d'un objet écouteur.
var nstepListener:Object = new Object();
nstepListener.change = function(evt_obj:Object){
 // evt_obj.target est le composant qui a généré l'événement change,
 // c'est-à-dire, l'incrémenteur numérique.
 trace("Value changed to " + evt_obj.target.value);
}
// Ajout de l'écouteur.
my_nstep.addEventListener("change", nstepListener);
```

NumericStepper.maximum

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

numericStepperInstance.maximum

Description

Propriété : la valeur de plage maximum de l'incrémenteur. Cette propriété peut contenir un nombre à trois chiffres. La valeur par défaut est 10.

Exemple

L'exemple suivant définit la valeur maximum de la plage de l'incrémenteur à 20.

Faites glisser une occurrence du composant NumericStepper sur la scène et nommez l'occurrence, `my_nstep` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant NumericStepper sur la scène (nom d'occurrence : my_nstep)
*/
var my_nstep:mx.controls.NumericStepper;

my_nstep.maximum = 20;
```

Voir aussi

NumericStepper.minimum

NumericStepper.minimum

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

numericStepperInstance.minimum

Description

Propriété : la valeur de plage minimum de l'incrémenteur. Cette propriété peut contenir un nombre à trois chiffres. La valeur par défaut est 0.

Exemple

L'exemple suivant définit la valeur minimum et la valeur initiale de l'occurrence de composant NumericStepper sur 100 et la valeur maximum sur 120.

Faites glisser une occurrence du composant NumericStepper sur la scène et nommez l'occurrence `my_nstep` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant NumericStepper sur la scène (nom d'occurrence : my_nstep)
*/

var my_nstep:mx.controls.NumericStepper;

my_nstep.minimum = 100;
my_nstep.maximum = 120;
my_nstep.value = my_nstep.minimum;
```

Voir aussi

NumericStepper.maximum

NumericStepper.nextValue

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

numericStepperInstance.nextValue

Description

Propriété (lecture seule) : la prochaine valeur séquentielle. Cette propriété peut contenir un nombre à trois chiffres.

Exemple

L'exemple suivant définit la valeur initiale de l'occurrence du composant NumericStepper sur -6 et la propriété `stepSize` sur 3. Il affiche ensuite la valeur de la propriété `nextValue` dans le panneau Sortie. La même valeur devrait s'afficher lorsque vous cliquez sur la flèche vers le haut sur l'incrémenteur.

Faites glisser une occurrence du composant NumericStepper sur la scène et nommez l'occurrence `my_nstep` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant NumericStepper sur la scène (nom d'occurrence : my_nstep)
*/
var my_nstep:mx.controls.NumericStepper;

my_nstep.stepSize = 3;
my_nstep.minimum = -6;
my_nstep.maximum = 12;
my_nstep.value = my_nstep.minimum;
trace(my_nstep.nextValue); // -3
```

Voir aussi

NumericStepper.previousValue

NumericStepper.previousValue

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

numericStepperInstance.previousValue

Description

Propriété (lecture seule) : la valeur séquentielle précédente. Cette propriété peut contenir un nombre à trois chiffres.

Exemple

L'exemple suivant définit la valeur initiale de l'occurrence NumericStepper sur la valeur minimale 6. Il définit la valeur `stepSize` sur 3 et crée un objet écouteur pour un événement change. Lorsqu'un événement change a lieu, l'exemple affiche la propriété `previousValue` dans le panneau Sortie.

Faites glisser une occurrence du composant NumericStepper sur la scène et nommez l'occurrence `my_nstep` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant NumericStepper sur la scène (nom d'occurrence : my_nstep)
*/

var my_nstep:mx.controls.NumericStepper;

my_nstep.minimum = 6;
my_nstep.value = my_nstep.minimum;
my_nstep.maximum = 120;
my_nstep.stepSize = 3;

// Création d'un objet écouteur.
var nstepListener:Object = new Object();
nstepListener.change = function(evt_obj:Object) {
 trace("previous value = " + evt_obj.target.previousValue);
}

// Ajout de l'écouteur.
my_nstep.addEventListener("change", nstepListener);
```

Voir aussi

NumericStepper.nextValue

NumericStepper.stepSize

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

numericStepperInstance.stepSize

Description

Propriété : la quantité d'unités à modifier à partir de la valeur courante. La valeur par défaut est 1. Cette valeur ne peut être 0. Cette propriété peut contenir un nombre à trois chiffres.

Exemple

L'exemple suivant définit la valeur initiale de l'occurrence NumericStepper sur la valeur minimale 3. Il définit également la valeur `stepSize` sur 3 pour que l'incrémenteur numérique incrémente de 3 lorsque l'utilisateur clique sur la flèche vers le haut et décrémente de 3 lorsque celui-ci clique sur la flèche vers le bas.

Faites glisser une occurrence du composant NumericStepper sur la scène et nommez l'occurrence `my_nstep` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant NumericStepper sur la scène (nom d'occurrence : my_nstep)
*/

var my_nstep:mx.controls.NumericStepper;

my_nstep.minimum = 3;
my_nstep.maximum = 120;
my_nstep.value = my_nstep.minimum;
my_nstep.stepSize = 3;
```

NumericStepper.value

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

numericStepperInstance.value

Description

Propriété : la valeur courante est affichée dans la zone de texte de l'incrémenteur. La valeur n'est pas affectée si elle ne correspond pas à la plage de l'incrémenteur et à l'unité d'incrémentation telles que définies dans la propriété `stepSize`. Cette propriété peut contenir un nombre à trois chiffres.

Exemple

L'exemple suivant définit la valeur actuelle de l'occurrence NumericStepper sur 10 et envoie la valeur vers le panneau Sortie.

Faites glisser une occurrence du composant NumericStepper sur la scène et nommez l'occurrence `my_nstep` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant NumericStepper sur la scène (nom d'occurrence : my_nstep)
*/

var my_nstep:mx.controls.NumericStepper;

my_nstep.value = 10;
my_nstep.maximum = 100;
trace(my_nstep.value); // 10
```

CHAPITRE 33

Classe PopUpManager

33

Nom de classe ActionScript mx.managers.PopUpManager

La classe PopUpManager vous permet de créer des fenêtres chevauchées qui peuvent être modales ou non modales. (Une fenêtre modale ne permet pas d'interagir avec d'autres fenêtres lorsqu'elle est active.) Vous utilisez les méthodes de cette classe pour créer et détruire des fenêtres contextuelles.

REMARQUE	La classe PopupManager est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Méthodes de la classe PopUpManager

Le tableau suivant répertorie les méthodes de la classe PopUpManager.

Méthode	Description
`PopUpManager.createPopUp()`	Crée une fenêtre contextuelle.
`PopUpManager.deletePopUp()`	Supprime une fenêtre contextuelle créée suite à un appel à `PopUpManager.createPopUp()`.

PopUpManager.createPopUp()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004 ou Flash MX Professional 2004.

Utilisation

```
PopUpManager.createPopUp(parent, class, modal [, initobj, outsideEvents])
```

Paramètres

parent Référence à une fenêtre par dessus laquelle afficher une fenêtre contextuelle.

class Référence à la classe de l'objet à créer.

modal Valeur booléenne indiquant si la fenêtre est modale (`true`) ou non (`false`).

initobj Objet contenant les propriétés d'initialisation. Ce paramètre est facultatif.

outsideEvents Valeur booléenne indiquant si un événement est déclenché lorsque l'utilisateur clique en dehors de la fenêtre (`true`) ou non (`false`). Ce paramètre est facultatif.

Valeur renvoyée

Une référence à l'objet créé.

Si le paramètre *class* est Window et qu'un composant de fenêtre se trouve dans la bibliothèque, la référence renvoyée est une fenêtre.

Description

Méthode : si elle est modale, un appel à `createPopUp()` trouve la fenêtre parent principale commençant par parent et crée une occurrence de la classe. Si elle est non modale, un appel à `createPopUp()` crée une occurrence de la classe comme enfant de la fenêtre parent.

Exemple

Le code suivant crée une fenêtre modale lorsque l'utilisateur clique sur le bouton :

```
lo = new Object();
lo.click = function(){
  mx.managers.PopUpManager.createPopUp(_root, mx.containers.Window, true);
}
button.addEventListener("click", lo);
```

PopUpManager.deletePopUp()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004 ou Flash MX Professional 2004

Utilisation

```
windowInstance.deletePopUp();
```

Paramètres

Aucun.

Valeur renvoyée

Aucune.

Description

Méthode : supprime une fenêtre contextuelle ainsi que l'état modal. C'est à la fenêtre chevauchée d'appeler `PopUpManager.deletePopUp()` lorsque la fenêtre est détruite.

Exemple

Le code suivant crée une fenêtre modale, appelée `win` et dotée d'un bouton de fermeture, et la supprime lorsqu'un utilisateur clique sur ce bouton :

```
import mx.managers.PopUpManager
import mx.containers.Window
win = PopUpManager.createPopUp(_root, Window, true, {closeButton:true});
lo = new Object();
lo.click = function(){
  win.deletePopUp();
}
win.addEventListener("click", lo);
```

CHAPITRE 34

Composant ProgressBar

34

Le composant ProgressBar affiche la progression du contenu en chargement. Le composant ProgressBar permet d'afficher le statut de chargement des images et des parties d'une application. Le processus de chargement peut être déterminé ou indéterminé. Une barre de progression *determinate* est une représentation linéaire de la progression d'une tâche dans le temps. Elle est utilisée lorsque la quantité de contenu à charger est connue. Une barre de progression *indeterminate* est utilisée lorsque la quantité de contenu à charger est inconnue. Vous pouvez ajouter une étiquette pour afficher la progression du contenu en chargement.

REMARQUE

Un composant ProgressBar est pris en charge pour ActionScript 2.0 et ActionScript 3.0. Ce document présente le composant de la version 2. Si vous utilisez celui de la version 3, reportez-vous à « Utilisation du composant ProgressBar » dans *Utilisation des composants ActionScript 3.0*.

Le composant ProgressBar contient un embout gauche, un embout droit et un rail de progression. Les embouts sont les extrémités de la barre de progression, l'endroit où le rail de progression se termine visuellement. L'aperçu en direct des occurrences de ProgressBar reflète les modifications apportées aux paramètres dans l'inspecteur des propriétés ou l'inspecteur des composants pendant la programmation. Les paramètres suivants sont reflétés dans l'aperçu en direct : conversion, direction, label, labelPlacement, mode et source.

Utilisation du composant ProgressBar

Une barre de progression vous permet d'afficher la progression du contenu pendant le chargement. Ces informations sont essentielles pour l'utilisateur lorsqu'il interagit avec une application.

Vous pouvez utiliser le composant ProgressBar dans plusieurs modes. Définissez le mode à l'aide du paramètre mode. Les modes les plus couramment utilisés sont « event » et « polled ». Ces méthodes utilisent le paramètre source pour spécifier un processus de chargement qui émet des événements `progress` et `complete` (mode event et polled) ou expose des méthodes `getBytesLoaded` et `getBytesTotal` (mode polled). Vous pouvez également utiliser le composant ProgressBar dans le mode manuel en définissant manuellement les propriétés `maximum`, `minimum` et `indeterminate` ainsi que les appels à la méthode `ProgressBar.setProgress()`.

Paramètres de ProgressBar

Dans l'inspecteur des propriétés ou l'inspecteur de composants (option de menu Fenêtre > Inspecteur de composants), vous pouvez définir les paramètres de création suivants pour chaque occurrence ProgressBar :

conversion Nombre pour diviser les valeurs %1 et %2 dans la chaîne de l'étiquette avant qu'elles ne soient affichées. La valeur par défaut est 1.

direction Direction dans laquelle avance la barre de progression. Cette valeur peut être `right` ou `left`, la valeur par défaut étant `right`.

label Texte indiquant la progression du chargement. Ce paramètre est une chaîne au format " %1 sur %2 chargés (%3%%) ". Dans cette chaîne, %1 est une balise d'emplacement pour les octets courants chargés, %2 est une balise d'emplacement pour le total des octets chargés et %3 est une balise d'emplacement pour le pourcentage de contenu chargé. Les caractères « %% » sont une balise d'emplacement pour le caractère « % ». Si une valeur pour %2 est inconnue, elle est remplacée par deux points d'interrogation (??). Si une valeur est undefined, l'étiquette ne s'affiche pas.

labelPlacement Position de l'étiquette par rapport à la barre de progression. Ce paramètre peut prendre l'une des valeurs suivantes : `top`, `bottom`, `left`, `right`, `center`. La valeur par défaut est `bottom`.

mode Mode dans lequel opère la barre de progression. Cette valeur peut être : `event`, `polled` ou `manual`. La valeur par défaut est `event`.

source Chaîne devant être convertie en objet représentant le nom d'occurrence de la source.

Et, dans l'inspecteur des composants, vous pouvez définir les paramètres supplémentaires suivants pour chaque occurrence du composant ProgressBar (Fenêtre > Inspecteur de composants) :

visible est une valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`). La valeur par défaut est `true`.

REMARQUE	Les propriétés minHeight et minWidth sont utilisées par des routines de dimensionnement internes. Elles sont définies dans UIObject et remplacées par différents composants selon vos besoins. Ces propriétés peuvent être utilisées pour créer un gestionnaire de dispositions personnalisées. Dans le cas contraire, la définition de ces propriétés dans l'inspecteur de composants n'a aucun effet visible.

Vous pouvez contrôler ces options et d'autres options du composant ProgressBar à l'aide des propriétés, méthodes et événements d'ActionScript. Pour plus d'informations, voir « Classe ProgressBar », à la page 1042.

Création d'une application avec le composant ProgressBar

La procédure suivante explique comment ajouter un composant ProgressBar à une application lors de la programmation. Dans cet exemple, la barre de progression est utilisée en mode event. En mode event, le contenu en cours de chargement doit émettre des événements `progress` et `complete` pour permettre à la barre d'afficher la progression. (Ces événements sont émis par le composant Loader. Pour plus d'informations, voir la section « Composant Loader », à la page 845.

Pour créer une application avec le composant ProgressBar en mode event :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser un composant ProgressBar du panneau Composants sur la scène.
3. Dans l'inspecteur des propriétés, procédez comme suit :
   - Entrez le nom d'occurrence **my_pb**.
   - Sélectionnez le paramètre de mode Event.
4. Faites glisser un composant Loader du panneau Composants jusqu'à la scène.
5. Dans l'inspecteur des propriétés, saisissez le nom d'occurrence **my_ldr**.
6. Sélectionnez la barre de progression sur la scène et, dans l'inspecteur des propriétés, entrez le paramètre source **my_ldr**.

7. Sélectionnez l'image 1 dans le scénario, ouvrez le panneau Actions et entrez le code suivant qui charge un fichier JPEG dans le composant Loader :

```
/**
 Requiert :
  - composant Loader sur la scène (nom d'occurrence : my_ldr)
  - Composant ProgressBar sur la scène (nom d'occurrence : my_pb)
*/

System.security.allowDomain("http://www.helpexamples.com");

var my_ldr:mx.controls.Loader;
var my_pb:mx.controls.ProgressBar;

my_pb.source = my_ldr;
my_ldr.autoLoad = false;
my_ldr.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";

// Lorsque autoLoad est défini sur false, le chargement ne commence
// que tant que la méthode load() n'a pas été appelée.
my_ldr.load();
```

8. Choisissez Contrôle> Tester l'animation.

Dans l'exemple suivant, la barre de progression est utilisée en mode polled. En mode polled, la barre de progression utilise les méthodes `getBytesLoaded()` et `getBytesTotal()` de l'objet source pour afficher la progression.

Pour créer une application avec le composant ProgressBar en mode polled :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser un composant ProgressBar du panneau Composants sur la scène.
3. Dans l'inspecteur des propriétés, saisissez le nom d'occurrence **my_pb**.
4. Sélectionnez l'image 1 dans le scénario, ouvrez le panneau Actions et entrez le code suivant qui crée un objet Sound nommé `my_sound` et appelle la méthode `loadSound()` pour charger un son dans l'objet Sound :

```
/**
 Requiert :
  - Composant ProgressBar sur la scène (nom d'occurrence : my_pb)
*/

System.security.allowDomain("http://www.helpexamples.com");

var my_pb:mx.controls.ProgressBar;

my_pb.mode = "polled";
my_pb.source = "my_sound";
```

```
var pbListener:Object = new Object();
pbListener.complete = function(evt_obj:Object) {
 trace("Sound loaded");
}
my_pb.addEventListener("complete", pbListener);

var my_sound:Sound = new Sound();
my_sound.loadSound("http://www.helpexamples.com/flash/sound/disco.mp3",
  true);
```

5. Sélectionnez Contrôle > Tester l'animation.

Dans l'exemple suivant, la barre de progression est utilisée en mode manual. En mode manual, vous devez définir les propriétés `maximum`, `minimum` et `indeterminate` conjointement avec la méthode `setProgress()` pour afficher la progression. Vous ne définissez pas la propriété `source` en mode manuel.

Pour créer une application avec le composant ProgressBar en mode manual :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser un composant ProgressBar du panneau Composants sur la scène.
3. Dans l'inspecteur des propriétés, procédez comme suit :
   - Entrez le nom d'occurrence **my_pb**.
   - Sélectionnez le paramètre de mode Manual.
4. Sélectionnez l'image 1 dans le scénario, ouvrez le panneau Actions et entrez le code suivant qui met à jour manuellement la barre de progression sur chaque téléchargement de fichier en appelant la méthode `setProgress()` :

```
for (var i:Number = 1; i <= total; i++){
  // Insertion du code pour charger le fichier.
  my_pb.setProgress(i, total);
}
```

5. Choisissez Contrôle> Tester l'animation.

Les deux exemples suivants sont supplémentaires.

Pour créer une application avec le composant ProgressBar en mode manual (exemple 2) :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser un composant Label sur la scène et nommez l'occurrence, **my_label**.
3. Faites glisser un composant ProgressBar sur la scène et nommez l'occurrence, **my_pb**.
4. Sélectionnez la barre de progression my_pb sur la scène et définissez le paramètre de mode du composant sur « manual » dans l'inspecteur des propriétés.

5. Sélectionnez l'image 1 du scénario et, dans le panneau Actions, ajoutez le code ActionScript suivant :

```
var feed_xml:XML = new XML();
feed_xml.onLoad = function(success:Boolean):Void {
  clearInterval(timer);
  my_label.text = "XML Loaded";
  my_pb.setProgress(feed_xml.getBytesLoaded(),
  feed_xml.getBytesTotal());
};
function updatePB(local_xml:XML):Void {
  my_pb.setProgress(local_xml.getBytesLoaded(),
  local_xml.getBytesTotal());
}
var timer:Number = setInterval(updatePB, 100, feed_xml);
feed_xml.load("http://www.helpexamples.com/flash/xml/menu.xml");
```

6. Choisissez Contrôle> Tester l'animation.

Pour créer une application avec le composant ProgressBar en mode manuel (exemple 3) :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser un composant ProgressBar sur la scène et nommez l'occurrence, **my_pb**.
3. Sélectionnez la barre de progression my_pb sur la scène et définissez le paramètre de mode du composant sur « manual » dans l'inspecteur des propriétés.
4. Sélectionnez l'image 1 du scénario et, dans le panneau Actions, ajoutez le code ActionScript suivant :

```
var img_mcl:MovieClipLoader = new MovieClipLoader();
var mclListener:Object = new Object();
mclListener.onLoadProgress = function(target_mc:MovieClip,
  numBytesLoaded:Number, numBytesTotal:Number) {
  my_pb.setProgress(numBytesLoaded, numBytesTotal);
};
mclListener.onLoadComplete = function(target_mc:MovieClip) {
//my_pb._visible = false;
};
img_mcl.addListener(mclListener);
this.createEmptyMovieClip("image_mc", 20);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  image_mc);
```

REMARQUE	Vous pouvez retirer les commentaires de la ligne //my_pb._visible = false si vous souhaitez masquer le composant après le chargement du contenu.

5. Sélectionnez Contrôle > Tester l'animation.

Personnalisation du composant ProgressBar

Vous pouvez transformer un composant ProgressBar horizontalement au cours de la programmation et lors de l'exécution. Lors de la programmation, sélectionnez le composant sur la scène et utilisez l'outil Transformer librement ou une commande Modification > Transformer. A l'exécution, utilisez `UIObject.setSize()`.

L'embout gauche, l'embout droit et le graphique du rail de la barre de progression ont une taille fixe. Lorsque vous redimensionnez une barre de progression, sa partie centrale est redimensionnée pour être contenue entre les deux embouts. Si une barre de progression est trop petite, le rendu risque d'être incorrect.

Utilisation de styles avec le composant ProgressBar

Vous pouvez définir des propriétés de style pour modifier l'aspect d'une occurrence de barre de progression. Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, consultez « Utilisation de styles pour personnaliser la couleur et le texte des composants » dans *Utilisation des composants ActionScript 2.0*.

Un composant ProgressBar prend en charge les styles suivants :

Style	Thème	Description
`themeColor`	Halo	Couleur de base d'un composant. `"haloGreen"`, `"haloBlue"` et `"haloOrange"` sont des valeurs possibles. La valeur par défaut est `"haloGreen"`.
`color`	Les deux	Couleur du texte. La valeur par défaut est 0x0B333C pour le thème Halo et vide pour le thème Sample.
`disabledColor`	Les deux	Couleur du texte lorsque le composant est désactivé. La couleur par défaut est 0x848384 (gris foncé).
`embedFonts`	Les deux	Valeur booléenne qui indique si la police spécifiée dans `fontFamily` est de type intégré. Ce style doit être défini sur `true` si `fontFamily` fait référence à une police intégrée. Dans le cas contraire, la police intégrée n'est pas utilisée. Si ce style est défini sur `true` et que `fontFamily` ne fait pas référence à une police intégrée, aucun texte n'est affiché. La valeur par défaut est `false`.
`fontFamily`	Les deux	Nom de la police du texte. La valeur par défaut est `"_sans"`.

Style	Thème	Description
`fontSize`	Les deux	Taille, en points, de la police. La valeur par défaut est 10.
`fontStyle`	Les deux	Style de police : `"normal"` ou `"italic"`. La valeur par défaut est `"normal"`.
`fontWeight`	Les deux	Epaisseur de la police : `"none"` ou `"bold"`. La valeur par défaut est `"none"`. Tous les composants peuvent également accepter la valeur `"normal"` au lieu de `"none"` pendant un appel à la méthode `setStyle()`, mais les appels suivants à `getStyle()` renvoient `"none"`.
`textDecoration`	Les deux	Décoration du texte : `"none"` ou `"underline"`. La valeur par défaut est `"none"`.
`barColor`	Sample	Couleur d'avant-plan à la fin du pourcentage. La couleur par défaut est le blanc. Pour définir la couleur de la barre sur un composant dont le thème est Halo, configurez la propriété de style `themeColor`.
`trackColor`	Sample	Couleur d'arrière-plan de la barre. La valeur par défaut est 0x666666 (gris foncé).

Utilisation d'enveloppes avec le composant ProgressBar

Le composant ProgressBar utilise des enveloppes pour représenter le rail de la barre de progression, la barre terminée et une barre indéterminée. Pour appliquer une enveloppe au composant ProgressBar lors de la programmation, modifiez les symboles dans le dossier Flash UI Components 2/Themes/MMDefault/ProgressBar Elements. Pour plus d'informations, reportez-vous à « A propos de l'application d'enveloppes aux composants » dans *Utilisation des composants ActionScript 2.0*.

Les graphiques de la barre et du rail sont constitués de trois enveloppes correspondant aux embouts gauche et droit et au centre. Les embouts sont utilisés « tels quels » et le centre est redimensionné horizontalement en fonction de la largeur de l'occurrence de ProgressBar.

La barre indeterminate est utilisée lorsque la propriété `indeterminate` de l'occurrence ProgressBar est définie sur `true`. L'enveloppe est redimensionnée horizontalement en fonction de la largeur de la barre de progression.

Le composant ProgressBar prend en charge les propriétés d'enveloppe suivantes :

Propriété	Description
`progTrackMiddleName`	Le milieu extensible du rail. La valeur par défaut est `ProgTrackMiddle`.
`progTrackLeftName`	L'embout gauche à taille fixe. La valeur par défaut est `ProgTrackLeft`.
`progTrackRightName`	L'embout droit à taille fixe. La valeur par défaut est `ProgTrackRight`.
`progBarMiddleName`	Le graphique central extensible de la barre. La valeur par défaut est `ProgBarMiddle`.
`progBarLeftName`	L'embout gauche de barre à taille fixe. La valeur par défaut est `ProgBarLeft`.
`progBarRightName`	L'embout droit de barre à taille fixe. La valeur par défaut est `ProgBarRight`.
`progIndBarName`	Le graphique de barre indéterminé. La valeur par défaut est `ProgIndBar`.

Pour créer des symboles de clip pour les enveloppes ProgressBar :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Choisissez Fichier > Importer > Ouvrir une bibliothèque externe et sélectionnez le fichier HaloTheme.fla.

   Ce fichier est stocké dans le dossier Configuration au niveau de l'application. Pour connaître son emplacement exact dans votre système d'exploitation, reportez-vous à « Présentation des thèmes » dans *Utilisation des composants ActionScript 2.0*.
3. Dans le panneau Bibliothèque du thème, choisissez Flash UI Components 2/Themes/MMDefault et faites glisser le dossier ProgressBar Assets dans la bibliothèque de votre document.
4. Développez le dossier ProgressBar Assets/States dans la bibliothèque de votre document.
5. Ouvrez les symboles que vous souhaitez personnaliser pour la modification.

   Par exemple, ouvrez le symbole ProgIndBar.
6. Personnalisez le symbole selon vos besoins.

   Par exemple, renversez le rail horizontalement.
7. Répétez les étapes 5 et 6 pour tous les symboles devant être personnalisés.

8. Cliquez sur le bouton Précédent pour revenir au scénario principal.
9. Faites glisser un composant ProgressBar sur la scène.

   Pour visualiser les enveloppes modifiées dans cet exemple, utilisez ActionScript pour définir la propriété `indeterminate` sur `true`.
10. Sélectionnez Contrôle > Tester l'animation.

Classe ProgressBar

Héritage MovieClip > Classe UIObject > ProgressBar

Nom de classe ActionScript mx.controls.ProgressBar

La définition d'une propriété de la classe ProgressBar avec ActionScript annule le paramètre du même nom défini dans l'inspecteur des propriétés ou des composants.

Toutes les classes de composants ont une propriété `version` qui correspond à une propriété de classe. Les propriétés de classe ne sont disponibles que pour la classe elle-même. La propriété `version` renvoie une chaîne qui indique la version du composant. Pour accéder à cette propriété, utilisez le code suivant :

```
trace(mx.controls.ProgressBar.version);
```

REMARQUE

Le code `trace(myProgressBarInstance.version);` renvoie `undefined`.

Méthodes de la classe ProgressBar

Le tableau suivant présente la méthode de la classe ProgressBar.

Méthode	Description
`ProgressBar.setProgress()`	Définit l'état de la barre de progression pour refléter le niveau de progression atteint lorsque la barre de progression est en mode manual

Méthodes héritées de la classe UIObject

Le tableau suivant répertorie les méthodes de la classe ProgressBar héritées de la classe UIObject. Pour appeler ces méthodes à partir de l'objet ProgressBar, utilisez le formulaire `ProgressBar.methodName`.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
`UIObject.setSize()`	Redimensionne l'objet à la taille demandée.
`UIObject.setSkin()`	Définit une enveloppe dans l'objet.
`UIObject.setStyle()`	Définit la propriété de style sur l'objet ou la déclaration de style.

Propriétés de la classe ProgressBar

Le tableau suivant répertorie les propriétés de la classe ProgressBar.

Propriété	Description
`ProgressBar.conversion`	Nombre utilisé pour convertir la valeur courante des octets chargés et les valeurs totales des octets chargés.
`ProgressBar.direction`	Direction dans laquelle la barre de progression se remplit.
`ProgressBar.indeterminate`	Indique si la taille de la source de chargement est inconnue.
`ProgressBar.label`	Texte accompagnant la barre de progression.
`ProgressBar.labelPlacement`	Emplacement de l'étiquette par rapport à la barre de progression.
`ProgressBar.maximum`	Valeur maximum de la barre de progression en mode manuel.

Propriété	Description
`ProgressBar.minimum`	Valeur minimum de la barre de progression en mode manuel.
`ProgressBar.mode`	Mode dans lequel la barre de progression charge le contenu.
`ProgressBar.percentComplete`	Lecture seule : nombre indiquant le pourcentage chargé.
`ProgressBar.source`	Contenu à charger.
`ProgressBar.value`	Lecture seule ; indique le niveau de progression atteint.

Propriétés héritées de la classe UIObject

Le tableau suivant répertorie les propriétés de la classe ProgressBar héritées de la classe UIObject. Pour appeler ces propriétés à partir de l'objet ProgressBar, utilisez le formulaire `ProgressBar.`*`propertyName`*.

Propriété	Description
`UIObject.bottom`	Position du bord inférieur de l'objet par rapport au bord inférieur de son parent. Lecture seule.
`UIObject.height`	Hauteur de l'objet, en pixels. Lecture seule.
`UIObject.left`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.right`	Position du bord droit de l'objet par rapport au bord droit de son parent. Lecture seule.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Position du bord supérieur de l'objet par rapport à son parent. Lecture seule.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Largeur de l'objet, en pixels. Lecture seule.
`UIObject.x`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.y`	Bord supérieur de l'objet, en pixels. Lecture seule.

Evénements de la classe ProgressBar

Le tableau suivant répertorie les événements de la classe ProgressBar.

Evénement	Description
`ProgressBar.complete`	Déclenché une fois le téléchargement terminé.
`ProgressBar.progress`	Déclenché pendant le chargement du contenu en mode manual ou polled.

Evénements hérités de la classe UIObject

Le tableau suivant répertorie les événements de la classe ProgressBar hérités de la classe UIObject. Pour appeler ces événements à partir de l'objet ProgressBar, utilisez le formulaire `ProgressBar.`*`eventName`*.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

ProgressBar.complete

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.complete = function(eventObject:Object) {
  // ...
};
progressBarInstance.addEventListener("complete", listenerObject);
```

Utilisation 2 :

```
on (complete) {
  // ...
}
```

Objet événement

Outre les propriétés d'objet événement standard, deux propriétés supplémentaires sont définies pour l'événement `ProgressBar.complete` : `current` (la valeur chargée qui est égale au total) et `total` (la valeur totale).

Description

Evénement : diffusé à l'ensemble des écouteurs enregistrés une fois la progression du chargement terminée.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence de composant (*`progressBarInstance`*) distribue un événement (ici, `complete`) qui est géré par une fonction (également appelée *gestionnaire*) dans un objet écouteur (*`listenerObject`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée au déclenchement de l'événement. Lorsqu'il est déclenché, l'événement transmet automatiquement un objet événement (*`eventObject`*) à la méthode de l'objet écouteur. Chaque objet événement est doté de propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence ProgressBar. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence du composant. Par exemple, le code suivant, associé à l'occurrence `progressBarInstance`, envoie « _level0.progressBarInstance » dans le panneau Sortie :

```
on (complete) {
  trace(this);
}
```

Exemple

Cet exemple crée un composant Loader, un composant ProgressBar (`my_pb`) pour ce dernier et un écouteur qui masque la barre de progression lorsque l'événement `complete` a lieu. L'exemple charge une image dans le composant Loader `my_ldr`.

Vous devez d'abord faire glisser un composant Loader et un composant ProgressBar depuis le panneau Composants vers la bibliothèque du document en cours, puis ajouter le code suivant à l'image 1 du scénario principal :

```
/**
 Requiert :
  - Composant ProgressBar dans la bibliothèque
  - Composant Loader dans la bibliothèque
*/

System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.controls.Loader, "my_ldr", 10, {autoLoad:false});
this.createClassObject(mx.controls.ProgressBar, "my_pb", 20,
  {indeterminate:true, source:my_ldr, mode:"polled"});

// Création d'un objet écouteur.
var pbListener:Object = new Object();
pbListener.complete = function(evt_obj:Object) {
 my_pb.visible = false;
};
// Ajout de l'écouteur.
my_pb.addEventListener("complete", pbListener);

my_ldr.load("http://www.helpexamples.com/flash/images/image2.jpg");
```

Voir aussi

`EventDispatcher.addEventListener()`

ProgressBar.conversion

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

progressBarInstance.conversion

Description

Propriété : nombre qui définit une valeur de conversion pour les valeurs entrantes. Il divise les valeurs courantes et totales, les réduit au minimum et affiche la valeur convertie dans la propriété `label`. La valeur par défaut est 1.

REMARQUE	La valeur minimum est le nombre entier le plus proche inférieur ou égal à la valeur spécifiée. Par exemple, 4,6 devient 4.

Exemple

Le code suivant affiche la progression du chargement d'un objet sound en divisant le nombre d'octets chargés par une valeur de conversion de 1024 afin de générer une valeur exprimée en kilo-octets.

Vous devez d'abord faire glisser un composant ProgressBar depuis le panneau Composants vers la bibliothèque du document en cours, puis ajouter le code suivant à l'image 1 du scénario principal :

```
/**
 Requiert :
  - Composant ProgressBar dans la bibliothèque
*/

System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.controls.ProgressBar, "my_pb", 20);

// Définition des attributs de la barre de progression.
my_pb.mode = "polled";
my_pb.source = "my_sound";
my_pb.label = "%1 kb loaded";
my_pb.conversion = 1024;

// Chargement du son.
var my_sound:Sound = new Sound();
my_sound.loadSound("http://www.helpexamples.com/flash/sound/disco.mp3", true);
```

ProgressBar.direction

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

progressBarInstance.direction

Description

Propriété : indique le sens dans lequel la barre de progression se remplit. Une valeur `right` indique que la barre se remplit de gauche à droite. Une valeur `left` indique que la barre se remplit de droite à gauche. La valeur par défaut est `right`.

Exemple

Le code suivant charge un objet sound et marque la progression avec une barre de progression qui se remplit vers la gauche.

Vous devez d'abord faire glisser un composant ProgressBar depuis le panneau Composants vers la bibliothèque du document en cours, puis ajouter le code suivant à l'image 1 du scénario principal :

```
/**
 Requiert :
  - Composant ProgressBar dans la bibliothèque
*/

System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.controls.ProgressBar, "my_pb", 20);

// Définition des attributs de la barre de progression.
my_pb.mode = "polled";
my_pb.source = "my_sound";
my_pb.direction = "left";

// Chargement du son.
var my_sound:Sound = new Sound();
my_sound.loadSound("http://www.helpexamples.com/flash/sound/disco.mp3",
  true);
```

ProgressBar.indeterminate

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

progressBarInstance.indeterminate

Description

Propriété : valeur booléenne indiquant si la barre de progression a un remplissage rayé et une source de chargement de taille inconnue (`true`), ou un remplissage uni et une source de chargement de taille connue (`false`). Par exemple, vous pouvez utiliser cette propriété si vous chargez un jeu de données volumineux dans un fichier SWF et que vous ignorez la taille des données que vous chargez.

Exemple

Le code suivant crée une barre de progression indéterminée avec un remplissage rayé qui se déplace de la gauche vers la droite :

Vous devez d'abord faire glisser un composant Loader et un composant ProgressBar depuis le panneau Composants jusqu'à la bibliothèque du document actif, puis ajouter le code suivant à l'image 1 du scénario principal :

```
/**
 Requiert :
  - Composant ProgressBar dans la bibliothèque
  - Composant Loader dans la bibliothèque
*/

System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.controls.ProgressBar, "my_pb", 10);
this.createClassObject(mx.controls.Loader, "my_ldr", 20);

// Création d'un objet écouteur.
var pbListener:Object = new Object();
pbListener.complete = function(evt_obj:Object) {
trace("Height: " + evt_obj.target.height + ", Width: " +
  evt_obj.target.width);};
// Ajout de l'écouteur.
my_pb.addEventListener("complete", pbListener);
```

```
// Définition des paramètres de la barre de progression.
my_pb.mode = "polled";
my_pb.indeterminate = true;
my_pb.source = my_ldr;

// Définition des paramètres du chargeur.
my_ldr.autoLoad = false;
my_ldr.scaleContent = false;
my_ldr.move(100, 100)
my_ldr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

ProgressBar.label

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

progressBarInstance.label

Description

Propriété : texte indiquant la progression du chargement. Cette propriété est une chaîne au format « `%1 sur %2 chargés (%3%%)` ». Dans cette chaîne, `%1` est une balise d'emplacement pour les octets courants chargés, `%2` est une balise d'emplacement pour le total des octets chargés et `%3` est une balise d'emplacement pour le pourcentage de contenu chargé. (Les caractères `%%` permettent à Flash d'afficher un seul caractère `%`.) Si une valeur pour `%2` est inconnue, elle est remplacée par `??`. Si une valeur est `undefined`, l'étiquette ne s'affiche pas. La valeur par défaut est « `LOADING %3%%` »

Exemple

L'exemple suivant charge une image dans un chargeur et marque la progression avec une barre de progression dont l'étiquette indique le pourcentage du nombre de kilo-octets total chargé.

Vous devez d'abord faire glisser un composant Loader et un composant ProgressBar depuis le panneau Composants vers la bibliothèque du document en cours, puis ajouter le code suivant à l'image 1 du scénario principal :

```
/**
 Requiert :
  - Composant ProgressBar dans la bibliothèque
  - Composant Loader dans la bibliothèque
*/

System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.controls.ProgressBar, "my_pb", 10);
this.createClassObject(mx.controls.Loader, "my_ldr", 20);

my_ldr.move(0, 30);

// Définition des paramètres de la barre de progression.
my_pb.mode = "polled";
my_pb.source = my_ldr;
my_pb.label = "%1 of %2 KB loaded";
my_pb.conversion = 1024; // 1024 octets dans un Ko

// Définition des paramètres du chargeur.
my_ldr.autoLoad = false;
my_ldr.scaleContent = false;
my_ldr.load("http://www.helpexamples.com/flash/images/image1.jpg")
```

Voir aussi

`ProgressBar.labelPlacement`

ProgressBar.labelPlacement

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`progressBarInstance``.labelPlacement`

Description

Propriété : définit le placement de l'étiquette par rapport à la barre de progression. Les valeurs possibles sont « `left` », « `right` », « `top` », « `bottom` » et « `center` ».

Exemple

L'exemple suivant charge une image dans un chargeur et marque la progression avec une barre de progression. Il définit la propriété `labelPlacement` sur `top` pour placer l'étiquette au-dessus de la barre de progression.

Vous devez d'abord faire glisser un composant Loader et un composant ProgressBar depuis le panneau Composants jusqu'à la bibliothèque du document actif, puis ajouter le code suivant à l'image 1 du scénario principal :

```
/**
 Requiert :
  - Composant ProgressBar dans la bibliothèque
  - Composant Loader dans la bibliothèque
*/

System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.controls.ProgressBar, "my_pb", 10);
this.createClassObject(mx.controls.Loader, "my_ldr", 20);

my_ldr.move(0, 30);

// Définition des paramètres de la barre de progression.
my_pb.mode = "polled";
my_pb.source = my_ldr;
my_pb.label = "%1 of %2 KB loaded";
my_pb.conversion = 1024; // 1024 octets dans un Ko
my_pb.labelPlacement = "top";

// Définition des paramètres du chargeur.
my_ldr.autoLoad = false;
my_ldr.scaleContent = false;
my_ldr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

Voir aussi

`ProgressBar.label`

ProgressBar.maximum

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

progressBarInstance.maximum

Description

Propriété ; la plus grande valeur de la barre de progression lorsque la propriété ProgressBar.mode est définie sur "manual".

Exemple

L'exemple suivant incrémente un composant ProgressBar manuellement jusqu'à une valeur maximum de 200 à laquelle il s'arrête. Il affiche l'incrément dans le panneau de sortie au fur et à mesure que la valeur augmente.

Faites glisser une occurrence du composant ProgressBar sur la scène et nommez l'occurrence, my_pb dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant ProgressBar sur la scène (nom d'occurrence : my_pb)
*/

var my_pb:mx.controls.ProgressBar;

// Définition du mode de la barre de progression.
my_pb.mode = "manual";
my_pb.label = "%1 out of %2 loaded";

// Valeur numérique minimale avant l'incrémentation de la barre
// de progression.
my_pb.minimum = 100;
```

```
// Valeur maximale de la barre de progression avant l'arrêt.
my_pb.maximum = 200;

var increment_num:Number = my_pb.minimum;
this.onEnterFrame = function() {
 if (increment_num < my_pb.maximum) {
  increment_num++;
  // Mise à jour de la progression du chiffre qui augmente.
  my_pb.setProgress(increment_num, my_pb.maximum);
  trace(increment_num);
 } else {
  delete this.onEnterFrame;
 }
};
```

Voir aussi

`ProgressBar.minimum`, `ProgressBar.mode`

ProgressBar.minimum

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`progressBarInstance``.minimum`

Description

Propriété ; la plus petite valeur de la barre de progression lorsque la propriété `ProgressBar.mode` est définie sur `"manual"`.

Exemple

L'exemple suivant incrémente manuellement un composant ProgressBar, en partant d'une valeur minimum de 100. Il affiche l'incrément dans le panneau de sortie au fur et à mesure que la valeur augmente.

Faites glisser une occurrence du composant ProgressBar sur la scène et nommez l'occurrence, `my_pb` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Composant ProgressBar sur la scène (nom d'occurrence : my_pb)
*/
```

```
var my_pb:mx.controls.ProgressBar;

// Définition du mode de la barre de progression.
my_pb.mode = "manual";
my_pb.label = "%1 out of %2 loaded";

// Valeur numérique minimale avant l'incrémentation de la barre
// de progression.
my_pb.minimum = 100;

// Valeur maximale de la barre de progression avant l'arrêt.
my_pb.maximum = 200;

var increment_num:Number = my_pb.minimum;
this.onEnterFrame = function() {
 if (increment_num < my_pb.maximum) {
  increment_num++;
  // Mise à jour de la progression du chiffre qui augmente.
  my_pb.setProgress(increment_num, my_pb.maximum);
  trace(increment_num);
 } else {
  delete this.onEnterFrame;
 }
};
```

Voir aussi

ProgressBar.maximum, ProgressBar.mode

ProgressBar.mode

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

progressBarInstance.mode

Description

Propriété : mode dans lequel la barre de progression charge le contenu. Les valeurs possibles sont `"event"`, `"polled"` ou `"manual"`.

Les modes event et polled sont les modes les plus courants. En mode event, la propriété `source` spécifie le contenu de chargement qui émet les événements `progress` et `complete` ; vous devez utiliser un objet Loader dans ce mode. En mode polled, la propriété `source` spécifie le contenu de chargement (tel qu'un objet MovieClip) qui expose des méthodes `getBytesLoaded()` et `getsBytesTotal()`. Tout objet qui expose ces méthodes peut être utilisé comme source en mode polled (y compris un objet personnalisé ou le scénario principal).

Vous pouvez également utiliser le composant ProgressBar en mode manual si vous définissez manuellement les propriétés `maximum`, `minimum` et `indeterminate` et que vous appelez la méthode `ProgressBar.setProgress()`.

Exemple

L'exemple suivant charge une image dans un chargeur et marque la progression du chargement avec une barre de progression définie sur le mode `event`. A la fin du chargement, un écouteur pour l'événement `complete` affiche le nom de l'objet loader.

Vous devez d'abord faire glisser un composant Loader et un composant ProgressBar depuis le panneau Composants jusqu'à la bibliothèque du document actif, puis ajouter le code suivant à l'image 1 du scénario principal :

```
/**
 Requiert :
  - Composant ProgressBar dans la bibliothèque
  - Composant Loader dans la bibliothèque
*/

System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.controls.ProgressBar, "my_pb", 10);
this.createClassObject(mx.controls.Loader, "my_ldr", 20);

// Création d'un objet écouteur.
var ldrListener:Object = new Object();
ldrListener.complete = function(evt_obj:Object) {
  trace("Event complete for " + evt_obj.target);
};
// Ajout de l'écouteur.
my_ldr.addEventListener("complete", ldrListener);

// Définition des paramètres de la barre de progression.
my_pb.mode = "event";
my_pb.indeterminate = true;
```

```
my_pb.source = my_ldr;

// Définition des paramètres du chargeur.
my_ldr.move(0,30);
my_ldr.autoLoad = false;
my_ldr.scaleContent = false;
my_ldr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

ProgressBar.percentComplete

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

progressBarInstance.percentComplete

Description

Propriété (lecture seule) : pourcentage de contenu ayant été chargé. Cette valeur est réduite au minimum. (La valeur minimum est le nombre entier le plus proche inférieur ou égal à la valeur spécifiée. Par exemple, 7,8 devient 7.) La formule suivante est utilisée pour calculer le pourcentage :

```
100 * (value - minimum) / (maximum - minimum)
```

Exemple

L'exemple suivant charge une image dans un chargeur associé avec une barre de progression. Un écouteur pour l'événement `progress` et un autre écouteur pour l'événement `complete` accèdent à la propriété `percentComplete` pour afficher le pourcentage de chargement effectué.

Faites glisser une occurrence du composant ProgressBar sur la scène et nommez l'occurrence, `my_pb` dans l'inspecteur des propriétés. Faites glisser une occurrence du composant Loader sur la scène et nommez l'occurrence, `my_ldr` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Occurrence du composant Loader sur la scène (nom d'occurrence : my_ldr)
  - Occurrence du composant ProgressBar sur la scène (nom d'occurrence :
  my_pb)
*/
```

```
System.security.allowDomain("http://www.helpexamples.com");

var my_ldr:mx.controls.Loader;
var my_pb:mx.controls.ProgressBar;

my_pb.mode = "polled";
my_pb.source = my_ldr;
my_ldr.autoLoad = false;

var pbListener:Object = new Object();
pbListener.progress = function(evt_obj:Object) {
 trace("progress = " + my_pb.percentComplete + "%");
}
pbListener.complete = function(evt_obj:Object) {
 trace("complete = " + my_pb.percentComplete + "%");
}
my_pb.addEventListener("progress", pbListener);
my_pb.addEventListener("complete", pbListener);

// Lorsque autoLoad est défini sur false, le chargement ne commence
// pas tant que la méthode load() n'a pas été appelée.
my_ldr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

ProgressBar.progress

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.progress = function(eventObject:Object) {
  // ...
};
progressBarInstance.addEventListener("progress", listenerObject);
```

Utilisation 2 :

```
on (progress) {
  // ...
}
```

Objet événement

Outre les propriétés d'objet événement standard, deux propriétés supplémentaires sont définies pour l'événement `ProgressBar.progress` : `current` (la valeur chargée qui est égale au `total`) et `total` (la valeur totale).

Description

Evénement : diffusé à l'ensemble des écouteurs enregistrés lorsque la valeur d'une barre de progression est modifiée.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence de composant (*`progressBarInstance`*) distribue un événement (ici, `progress`) qui est géré par une fonction (également appelée *gestionnaire*) dans un objet écouteur (*`listenerObject`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée au déclenchement de l'événement. Lorsqu'il est déclenché, l'événement transmet automatiquement un objet événement (*`eventObject`*) à la méthode de l'objet écouteur. Chaque objet événement est doté de propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence ProgressBar. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence du composant. Par exemple, le code suivant, associé à l'occurrence `progressBarInstance`, envoie « _level0.`progressBarInstance` » dans le panneau Sortie :

```
on (progress) {
  trace(this);
}
```

Exemple

Cet exemple charge une image dans un chargeur avec une barre de progression associée et crée un écouteur pour l'événement `progress`. Lorsque l'événement progress a lieu, l'exemple affiche la propriété `value` qui est une valeur comprise entre `ProgressBar.minimum` et `ProgressBar.maximum`.

Faites glisser une occurrence du composant ProgressBar sur la scène et nommez l'occurrence, `my_pb` dans l'inspecteur des propriétés. Faites glisser une occurrence du composant Loader sur la scène et nommez l'occurrence, `my_ldr` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Occurrence du composant Loader sur la scène (nom d'occurrence : my_ldr)
  - Occurrence du composant ProgressBar sur la scène (nom d'occurrence :
  my_pb)
*/

System.security.allowDomain("http://www.helpexamples.com");

var my_ldr:mx.controls.Loader;
var my_pb:mx.controls.ProgressBar;

my_pb.mode = "polled";
my_pb.source = my_ldr;
my_ldr.autoLoad = false;

// Création d'un objet écouteur.
var pbListener:Object = new Object();
pbListener.progress = function(evt_obj:Object) {
  // evt_obj.target est le composant qui a généré l'événement progress,
  // c'est-à-dire la barre de progression.
  trace("Current progress value = " + evt_obj.target.value);
};
// Ajout de l'écouteur.
my_pb.addEventListener("progress", pbListener);

// Lorsque autoLoad est défini sur false, le chargement ne commence
// pas tant que la méthode load() n'a pas été appelée.
my_ldr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

Voir aussi

`EventDispatcher.addEventListener()`

ProgressBar.setProgress()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

progressBarInstance.setProgress(*completed*, *total*)

Paramètres

completed Nombre indiquant le niveau de progression atteint. Vous pouvez utiliser les propriétés ProgressBar.label et ProgressBar.conversion pour afficher le nombre sous forme de pourcentage ou dans l'unité de votre choix, en fonction de la source de la barre de progression.

total Nombre indiquant la progression totale devant être effectuée pour atteindre 100 %.

Valeur renvoyée

Un nombre indiquant le niveau de progression atteint.

Description

Méthode ; définit l'état de la barre de progression pour refléter la progression effectuée lorsque la propriété ProgressBar.mode est définie sur "manual". Vous pouvez appeler cette méthode pour que la barre reflète l'état d'un processus autre que le chargement. Par exemple, vous pouvez définir explicitement la barre de progression sur une progression de zéro.

Le paramètre *completed* est affecté à la propriété value et le paramètre *total* est affecté à la propriété maximum. La propriété minimum n'est pas modifiée.

Exemple

L'exemple suivant définit le mode de la barre de progression sur manual et appelle setProgress() de la fonction onEnterFrame() qui est appelée de façon répétée à la fréquence d'images du fichier SWF. L'exemple définit la valeur minimum pour la barre de progression sur 100 et la valeur maximum sur 200. Il marque la progression par incréments de 1.

Faites glisser une occurrence du composant ProgressBar sur la scène et nommez l'occurrence, `my_pb` dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - ProgressBar sur la scène (nom d'occurrence : my_pb)
*/

var my_pb:mx.controls.ProgressBar;

// Définition du mode de la barre de progression.
my_pb.mode = "manual";
my_pb.label = "%1 out of %2 loaded";

// Valeur numérique minimale avant l'incrémentation de la barre
// de progression.
my_pb.minimum = 100;

// Valeur maximale de la barre de progression avant l'arrêt.
my_pb.maximum = 200;

var increment_num:Number = my_pb.minimum;
this.onEnterFrame = function() {
 if (increment_num < my_pb.maximum) {
  increment_num++;
  // Mise à jour de la progression du chiffre qui augmente.
  my_pb.setProgress(increment_num, my_pb.maximum);
  trace(increment_num);
 } else {
  delete this.onEnterFrame;
 }
};
```

ProgressBar.source

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

progressBarInstance.source

Description

Propriété : référence à l'occurrence devant être chargée et dont le processus de chargement sera affiché. Le contenu en chargement doit émettre un événement `progress` à partir duquel les valeurs courantes et totales sont récupérées. Cette propriété est uniquement utilisée lorsque `ProgressBar.mode` est défini sur `event` ou `polled`. La valeur par défaut est `undefined`.

Vous pouvez utiliser le composant ProgressBar avec du contenu dans une application, y compris `_root`.

Exemple

L'exemple suivant charge une image dans un chargeur et marque la progression avec une barre de progression. L'exemple définit la propriété `source` sur le nom du composant Loader (`my_ldr`) pour associer le contenu avec la barre de progression.

Vous devez d'abord faire glisser un composant Loader et un composant ProgressBar depuis le panneau Composants jusqu'à la bibliothèque du document actif, puis ajouter le code suivant à l'image 1 du scénario principal :

```
/**
 Requiert :
  - Composant ProgressBar dans la bibliothèque
  - Composant Loader dans la bibliothèque
*/

System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.controls.ProgressBar, "my_pb", 10);
this.createClassObject(mx.controls.Loader, "my_ldr", 20);

// Création d'un objet écouteur.
var pbListener:Object = new Object();
pbListener.complete = function(evt_obj:Object) {
 evt_obj.target.visible = false;
};
// Ajout de l'écouteur.
my_pb.addEventListener("complete", pbListener);

// Définition des paramètres de la barre de progression.
my_pb.mode = "polled";
my_pb.indeterminate = true;
my_pb.source = my_ldr;

// Définition des paramètres du chargeur.
my_ldr.autoLoad = false;
my_ldr.scaleContent = false;
my_ldr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

Voir aussi

`ProgressBar.mode`

ProgressBar.value

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

progressBarInstance.value

Description

Propriété (lecture seule) : indique le niveau de progression atteint. Cette propriété est un nombre compris entre la valeur ProgressBar.minimum et ProgressBar.maximum.
La valeur par défaut est 0.

Exemple

L'exemple suivant charge une image dans un composant Loader et marque la progression avec une barre de progression. A la fin du chargement, l'exemple affiche les valeurs minimum, maximum et actuelle pour la barre de progression.

Faites glisser une occurrence du composant ProgressBar sur la scène et nommez l'occurrence, my_pb dans l'inspecteur des propriétés. Faites glisser une occurrence du composant Loader sur la scène et nommez l'occurrence, my_ldr dans l'inspecteur des propriétés. Ajoutez le code suivant à l'image 1 du scénario :

```
/**
 Requiert :
  - Occurrence du composant Loader sur la scène (nom d'occurrence : my_ldr)
  - Occurrence du composant ProgressBar sur la scène (nom d'occurrence :
  my_pb)
*/

System.security.allowDomain("http://www.helpexamples.com");

var my_ldr:mx.controls.Loader;
var my_pb:mx.controls.ProgressBar;

my_pb.mode = "polled";
my_pb.source = my_ldr;
my_ldr.autoLoad = false;


// Création d'un objet écouteur.
var pbListener:Object = new Object();
```

```
pbListener.complete = function(evt_obj:Object){
  // event_obj.target est le composant qui a généré l'événement complete,
  // c'est-à-dire la barre de progression.
  trace("Minimum value is: " + evt_obj.target.minimum + " bytes");
  trace("Maximum value is: " + evt_obj.target.maximum + " bytes");
  trace("Current ProgressBar value = " + evt_obj.target.value + " bytes");
}
// Ajout de l'écouteur.
my_pb.addEventListener("complete", pbListener);

// Lorsque autoLoad est défini sur false, le chargement ne commence pas
// tant que la méthode load() n'a pas été appelée.
my_ldr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

CHAPITRE 35

35 Composant RadioButton

Le composant RadioButton vous permet d'obliger un utilisateur à faire un choix unique parmi plusieurs possibilités. Ce composant doit être utilisé dans un groupe comprenant au moins deux occurrences de RadioButton. Seul un membre peut être sélectionné au sein du groupe. La sélection d'un bouton radio dans un groupe désélectionne le bouton jusqu'alors sélectionné dans le groupe. Vous définissez le paramètre `groupName` pour indiquer le groupe auquel appartient un bouton radio.

REMARQUE Un composant RadioButton est pris en charge pour ActionScript 2.0 et ActionScript 3.0. Ce document présente le composant de la version 2. Si vous utilisez celui de la version 3, reportez-vous à « Utilisation du composant RadioButton » dans *Utilisation des composants ActionScript 3.0*.

Un bouton radio peut être activé ou désactivé. En état désactivé, le bouton radio ne réagit pas aux commandes de la souris ou du clavier. Lorsque l'utilisateur clique ou appuie sur la touche de tabulation pour ouvrir un groupe de composant RadioButton, seul le bouton radio sélectionné reçoit le focus. L'utilisateur peut ensuite utiliser les touches suivantes pour le contrôler :

Touche	Description
Flèche vers le haut/ Flèche vers la gauche	La sélection se déplace vers le bouton radio précédent dans le groupe de boutons radio.
Flèche vers le bas/ Flèche vers la droite	La sélection se déplace vers le bouton radio suivant dans le groupe de boutons radio.
Tab	Déplace le focus du groupe de boutons radio vers le composant suivant.

Pour plus d'informations sur le contrôle du focus, reportez-vous à « Classe FocusManager », à la page 745 ou à « Création de la navigation personnalisée du focus » dans *Utilisation des composants ActionScript 2.0*.

Un aperçu en direct de chaque occurrence de RadioButton sur la scène reflète les modifications effectuées sur les paramètres dans l'inspecteur Propriétés ou des composants lors de la programmation. Cependant, l'exclusion mutuelle de la sélection ne s'affiche pas dans l'aperçu en direct. Si vous définissez le paramètre sélectionné sur true pour deux boutons radios dans le même groupe, ils apparaissent tous deux comme étant sélectionnés même si seule la dernière occurrence créée apparaît comme étant sélectionnée lors de l'exécution. Pour plus d'informations, voir « Paramètres de RadioButton », à la page 1068.

Lorsque vous ajoutez le composant RadioButton à une application, vous pouvez utiliser le panneau Accessibilité pour le rendre accessible aux lecteurs d'écran. Commencez par ajouter la ligne de code suivante pour activer l'accessibilité :

```
mx.accessibility.RadioButtonAccImpl.enableAccessibility();
```

Quel que soit le nombre d'occurrences d'un composant, l'activation de son accessibilité ne se fait qu'une fois. Pour plus d'informations, reportez-vous à *Création de contenu accessible* dans *Utilisation de Flash*.

Utilisation du composant RadioButton

Un bouton radio est une partie essentielle de tout formulaire ou application Web. Vous pouvez utiliser les boutons radio partout où vous souhaitez qu'un utilisateur opte pour un choix dans un groupe d'options. Par exemple, utilisez des boutons radio dans un formulaire pour demander quelle carte bancaire un utilisateur souhaite utiliser.

Paramètres de RadioButton

Vous pouvez définir les paramètres de programmation suivants pour chaque occurrence du composant RadioButton dans l'inspecteur Propriétés ou des composants :

data correspond à la valeur associée au bouton radio. Il n'y a pas de valeur par défaut.

groupName est le nom du groupe du bouton radio. La valeur par défaut est `radioGroup`.

label définit la valeur du texte sur le bouton. La valeur par défaut est `Radio Button`.

labelPlacement oriente le texte d'étiquette sur le bouton. Ce paramètre peut prendre l'une des quatre valeurs suivantes : `left`, `right`, `top` ou `bottom`. La valeur par défaut est `right`. Pour plus d'informations, reportez-vous à `RadioButton.labelPlacement`.

selected active la valeur initiale du bouton radio (`true`) ou non (`false`). Un bouton radio sélectionné affiche un point. Seul un bouton radio dans un groupe peut avoir une valeur selected définie sur `true`. Si, dans un groupe, plusieurs boutons radio sont définis sur `true`, le dernier bouton radio à être instancié est sélectionné. La valeur par défaut est `false`.

ActionScript vous permet de définir des options supplémentaires pour les occurrences de RadioButton à l'aide des méthodes, propriétés et événements de la classe RadioButton. Pour plus d'informations, voir « Classe RadioButton », à la page 1074.

Création d'une application avec le composant RadioButton

La procédure suivante explique comment ajouter des composants RadioButton à une application lors de la programmation. Dans cet exemple, les boutons radio sont utilisés pour présenter une question à laquelle on ne peut répondre que par oui ou par non, « Are you a Flashist? ». Les données du groupe radio sont affichées dans un composant TextArea avec `theVerdict` comme nom d'occurrence.

Pour créer une application avec le composant RadioButton :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser deux composants RadioButton du panneau Composants jusqu'à la scène.
3. Sélectionnez l'un des boutons radio. Dans l'inspecteur des composants, procédez comme suit :
   - Entrez **Yes** pour le paramètre label.
   - Entrez **Flashist** pour le paramètre data.
4. Sélectionnez l'autre bouton radio. Dans l'inspecteur des composants, procédez comme suit :
   - Entrez **No** pour le paramètre label.
   - Entrez **Anti-Flashist** pour le paramètre data.
5. Faites glisser un composant TextArea du panneau Composants vers la scène et nommez l'occurrence `theVerdict`.
6. Sélectionnez l'image 1 dans le scénario principal, ouvrez le panneau Actions et saisissez le code suivant :

```
flashistListener = new Object();
flashistListener.click = function (evt){
    theVerdict.text = evt.target.selection.data
}
radioGroup.addEventListener("click", flashistListener);
```

   La dernière ligne de code ajoute un gestionnaire d'événement `click` au groupe de boutons radio `radioGroup`. Le gestionnaire définit la propriété `text` de `theVerdict` (une occurrence de TextArea) à la valeur de la propriété `data` du bouton radio sélectionné dans le groupe de boutons radio `radioGroup`. Pour plus d'informations, reportez-vous à `RadioButton.click`.

Personnalisation du composant RadioButton

Vous pouvez transformer un composant RadioButton horizontalement et verticalement au cours de la programmation et lors de l'exécution. Lors de la programmation, sélectionnez le composant sur la scène et utilisez l'outil Transformer librement ou une commande Modification > Transformer. A l'exécution, utilisez la méthode `setSize()` (voir `UIObject.setSize()`).

Le cadre de sélection d'un composant RadioButton est invisible et désigne également la zone active du composant. Si vous augmentez la taille du composant, vous augmentez également la taille de la zone active.

Si la dimension du cadre de sélection du composant est trop petite pour l'étiquette du composant, celle-ci sera rognée.

Utilisation de styles avec le composant RadioButton

Vous pouvez définir les propriétés des styles afin de modifier l'apparence d'un bouton radio. Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, consultez « Utilisation de styles pour personnaliser la couleur et le texte des composants » dans *Utilisation des composants ActionScript 2.0*.

Un composant RadioButton utilise les styles suivants :

Style	Thème	Description
`themeColor`	Halo	Couleur de base d'un composant. `"haloGreen"`, `"haloBlue"` et `"haloOrange"` sont des valeurs possibles. La valeur par défaut est `"haloGreen"`.
`color`	Les deux	Couleur du texte. La valeur par défaut est 0x0B333C pour le thème Halo et vide pour le thème Sample.
`disabledColor`	Les deux	Couleur du texte lorsque le composant est désactivé. La couleur par défaut est 0x848384 (gris foncé).

Style	Thème	Description
`embedFonts`	Les deux	Valeur booléenne qui indique si la police spécifiée dans `fontFamily` est de type intégré. Ce style doit être défini sur `true` si `fontFamily` fait référence à une police intégrée. Dans le cas contraire, la police intégrée n'est pas utilisée. Si ce style est défini sur `true` et que `fontFamily` ne fait pas référence à une police intégrée, aucun texte n'est affiché. La valeur par défaut est `false`.
`fontFamily`	Les deux	Nom de la police du texte. La valeur par défaut est `"_sans"`.
`fontSize`	Les deux	Taille, en points, de la police. La valeur par défaut est 10.
`fontStyle`	Les deux	Style de police : `"normal"` ou `"italic"`. La valeur par défaut est `"normal"`.
`fontWeight`	Les deux	Epaisseur de la police : `"none"` ou `"bold"`. La valeur par défaut est `"none"`. Tous les composants peuvent également accepter la valeur « `normal` » au lieu de « `none` » pendant un appel de `setStyle()`, mais les prochains appels de `getStyle()` renvoient « `none` ».
`textDecoration`	Les deux	Décoration du texte : `"none"` ou `"underline"`. La valeur par défaut est `"none"`.
`symbolBackgroundColor`	Sample	Couleur d'arrière-plan du bouton radio. La valeur par défaut est 0xFFFFFF (blanc).
`symbolBackgroundDisabledColor`	Sample	Couleur d'arrière-plan du bouton radio lorsqu'il est désactivé. La valeur par défaut est 0xEFEEEF (gris clair).
`symbolBackgroundPressedColor`	Sample	Couleur d'arrière-plan du bouton radio lorsqu'il est enfoncé. La valeur par défaut est 0xFFFFFF (blanc).
`symbolColor`	Sample	Couleur du point dans le bouton radio. La valeur par défaut est 0x000000 (noir).
`symbolDisabledColor`	Sample	Couleur du point dans le bouton radio lorsque le composant est désactivé. La valeur par défaut est 0x848384 (gris foncé).

Utilisation d'enveloppes avec le composant RadioButton

Vous pouvez appliquer une enveloppe au composant RadioButton au cours de la programmation en modifiant ses symboles dans la bibliothèque. Les enveloppes du composant RadioButton se trouvent dans le dossier suivant de la bibliothèque de HaloTheme.fla ou SampleTheme.fla : Flash UI Components 2/Themes/MMDefault/RadioButton Assets/States. Pour plus d'informations, consultez « A propos de l'application d'enveloppes aux composants » dans *Utilisation des composants ActionScript 2.0*.

Si un bouton radio est activé mais n'est pas sélectionné, il affiche son état survolé lorsque l'utilisateur place le pointeur de la souris au-dessus du bouton. Lorsque l'utilisateur clique sur un bouton radio non sélectionné, le bouton radio reçoit le focus d'entrée et affiche son état false enfoncé. Lorsque l'utilisateur relâche le bouton de la souris, le bouton radio affiche son état true et le bouton radio sélectionné précédemment dans le groupe revient à son état false. Si l'utilisateur éloigne le pointeur d'un bouton radio tout en appuyant sur le bouton de la souris, l'apparence du bouton revient à son état false et conserve le focus d'entrée.

Si un bouton radio ou un groupe de boutons radio est désactivé, il affiche l'état désactivé, quelle que soit l'interaction de l'utilisateur.

Un composant RadioButton utilise les propriétés d'enveloppe suivantes :

Nom	Description
`falseUpIcon`	Etat non sélectionné. La valeur par défaut est `RadioFalseUp`.
`falseDownIcon`	Etat enfoncé-non sélectionné. La valeur par défaut est `RadioFalseDown`.
`falseOverIcon`	Etat survolé-non sélectionné. La valeur par défaut est `RadioFalseOver`.
`falseDisabledIcon`	Etat désactivé-non sélectionné. La valeur par défaut est `RadioFalseDisabled`.
`trueUpIcon`	Etat sélectionné. La valeur par défaut est `RadioTrueUp`.
`trueDisabledIcon`	Etat désactivé-sélectionné. La valeur par défaut est `RadioTrueDisabled`.

Chacune de ces enveloppes correspond à l'icône qui signale l'état de RadioButton. Le composant RadioButton ne présente pas de bordure ni d'arrière-plan.

Pour créer des symboles de clip pour les enveloppes de RadioButton :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Choisissez Fichier > Importer > Ouvrir une bibliothèque externe et sélectionnez le fichier HaloTheme.fla.

   Ce fichier est stocké dans le dossier Configuration au niveau de l'application. Pour connaître son emplacement exact dans votre système d'exploitation, reportez-vous à « Présentation des thèmes » dans *Utilisation des composants ActionScript 2.0*.
3. Dans le panneau Bibliothèque du thème, développez le dossier Flash UI Components 2/Themes/MMDefault et faites glisser le dossier RadioButton Assets dans la bibliothèque de votre document.
4. Développez le dossier RadioButton Assets/States dans la bibliothèque de votre document.
5. Ouvrez les symboles que vous souhaitez personnaliser pour la modification.

   Par exemple, ouvrez le symbole RadioFalseDisabled.
6. Personnalisez le symbole selon vos besoins.

   Par exemple, appliquez la couleur gris clair au cercle intérieur blanc.
7. Répétez les étapes 5 et 6 pour tous les symboles devant être personnalisés.

   Par exemple, changez également la couleur pour le cercle intérieur du symbole RadioTrueDisabled.
8. Cliquez sur le bouton Précédent pour revenir au scénario principal.
9. Faites glisser un composant RadioButton sur la scène.

   Pour cet exemple, faites glisser deux occurrences afin d'illustrer les deux nouveaux symboles d'enveloppe.
10. Définissez les propriétés de l'occurrence de RadioButton comme nécessaire.

    Pour cet exemple, définissez une occurrence RadioButton sur selected et utilisez ActionScript pour définir les deux occurrences RadioButton sur disabled.
11. Sélectionnez Contrôle > Tester l'animation.

Classe RadioButton

Héritage MovieClip > Classe UIObject > Classe UIComponent > Classe SimpleButton > Composant Button > RadioButton

Nom du logiciel ActionScript mx.controls.RadioButton

Les propriétés de la classe RadioButton vous permettent de créer une étiquette de texte, lors de l'exécution, et de la disposer par rapport au bouton radio. Vous pouvez aussi affecter des valeurs de données aux boutons radio, les affecter à des groupes et les sélectionner en fonction de la valeur de données ou du nom de l'occurrence.

La définition d'une propriété de la classe RadioButton avec ActionScript annule le paramètre du même nom défini dans l'inspecteur Propriétés ou des composants.

Le composant RadioButton utilise le gestionnaire de focus pour remplacer le rectangle de focus par défaut de Flash Player et tracer un rectangle de focus personnalisé aux coins arrondis. Pour plus d'informations sur la création de la navigation du focus, reportez-vous à « Création de la navigation personnalisée du focus » dans *Utilisation des composants ActionScript 2.0*.

Toutes les classes de composants ont une propriété `version` qui correspond à une propriété de classe. Les propriétés de classe ne sont disponibles que pour la classe elle-même. La propriété `version` renvoie une chaîne qui indique la version du composant. Pour accéder à cette propriété, utilisez le code suivant :

```
trace(mx.controls.RadioButton.version);
```

REMARQUE	Le code `trace(myRadioButtonInstance.version);` renvoie `undefined`.

Méthodes de la classe RadioButton

La classe RadioButton ne présente pas de méthodes exclusives.

Méthodes héritées de la classe UIObject

Le tableau suivant répertorie les méthodes de la classe RadioButton héritées de la classe UIObject. Pour appeler ces méthodes à partir de l'objet RadioButton, utilisez le formulaire *RadioButtonInstance.methodName*.

Méthode	Description
UIObject.createClassObject()	Crée un objet dans la classe spécifiée.
UIObject.createObject()	Crée un sous-objet dans un objet.
UIObject.destroyObject()	Détruit une occurrence de composant.
UIObject.doLater()	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
UIObject.getStyle()	Obtient la propriété de style de l'objet ou de la déclaration de style.
UIObject.invalidate()	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
UIObject.move()	Déplace l'objet à l'emplacement demandé.
UIObject.redraw()	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
UIObject.setSize()	Redimensionne l'objet à la taille demandée.
UIObject.setSkin()	Définit une enveloppe dans l'objet.
UIObject.setStyle()	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant répertorie les méthodes de RadioButton héritées de la classe UIComponent. Pour appeler ces méthodes à partir de l'objet RadioButton, utilisez le formulaire *RadioButtonInstance.methodName*.

Méthode	Description
UIComponent.getFocus()	Renvoie une référence à l'objet ayant le focus.
UIComponent.setFocus()	Attribue le focus à l'occurrence de composant.

Propriétés de la classe RadioButton

Le tableau suivant répertorie les propriétés de la classe RadioButton.

Propriété	Description
`RadioButton.data`	Valeur associée à une occurrence de bouton radio.
`RadioButton.groupName`	Nom d'un groupe d'une occurrence de groupe de boutons radio ou d'une occurrence de bouton radio.
`RadioButton.label`	Texte affiché à côté d'un bouton radio.
`RadioButton.labelPlacement`	Orientation du texte de l'étiquette par rapport à un bouton radio ou à un groupe de boutons radio.
`RadioButton.selected`	Sélectionne le bouton radio et désélectionne le bouton radio sélectionné précédemment. Cette propriété peut être utilisée avec une occurrence de RadioButton ou de RadioButtonGroup.
`RadioButton.selectedData`	Sélectionne le bouton radio avec la valeur de données spécifiée dans un groupe de boutons radio.
`RadioButton.selection`	Référence au bouton radio actuellement sélectionné dans un groupe de boutons radio. Cette propriété peut être utilisée avec une occurrence de RadioButton ou de RadioButtonGroup.

Propriétés héritées de la classe UIObject

Le tableau suivant répertorie les propriétés de la classe RadioButton héritées de la classe UIObject. Pour appeler ces propriétés à partir de l'objet RadioButton, utilisez le formulaire *`RadioButtonInstance.propertyName`*.

Propriété	Description
`UIObject.bottom`	Position du bord inférieur de l'objet par rapport au bord inférieur de son parent. Lecture seule.
`UIObject.height`	Hauteur de l'objet, en pixels. Lecture seule.
`UIObject.left`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.right`	Position du bord droit de l'objet par rapport au bord droit de son parent. Lecture seule.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.

Propriété	Description
UIObject.top	Position du bord supérieur de l'objet par rapport à son parent. Lecture seule.
UIObject.visible	Valeur booléenne indiquant si l'objet est visible (true) ou non (false).
UIObject.width	Largeur de l'objet, en pixels. Lecture seule.
UIObject.x	Bord gauche de l'objet, en pixels. Lecture seule.
UIObject.y	Bord supérieur de l'objet, en pixels. Lecture seule.

Propriétés héritées de la classe UIComponent

Le tableau suivant répertorie les propriétés de la classe RadioButton héritées de la classe UIComponent. Pour appeler ces propriétés à partir de l'objet RadioButton, utilisez le formulaire *RadioButtonInstance.propertyName*.

Propriété	Description
UIComponent.enabled	Indique si le composant peut recevoir le focus et la saisie.
UIComponent.tabIndex	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Propriétés héritées de la classe SimpleButton

Le tableau suivant répertorie les propriétés de la classe RadioButton héritées de la classe SimpleButton. Pour appeler ces propriétés à partir de l'objet RadioButton, utilisez le formulaire *RadioButtonInstance.propertyName*.

Propriété	Description
SimpleButton.emphasized	Indique si un bouton a l'aspect d'un bouton-poussoir par défaut.
SimpleButton.emphasizedStyleDeclaration	Déclaration de style lorsque la propriété emphasized est définie sur true.
SimpleButton.selected	Valeur booléenne indiquant si le bouton est sélectionné (true) ou non (false). La valeur par défaut est false.
SimpleButton.toggle	Valeur booléenne indiquant si le comportement du bouton est celui d'un bouton bascule (true) ou non (false). La valeur par défaut est false.

Propriétés héritées de la classe Button

Le tableau suivant répertorie les propriétés de la classe RadioButton héritées de la classe Button. Pour appeler ces propriétés à partir de l'objet RadioButton, utilisez le formulaire *RadioButtonInstance.propertyName*.

Propriété	Description
Button.icon	Spécifie une icône pour une occurrence de bouton.
Button.label	Spécifie le texte qui apparaît dans un bouton.
Button.labelPlacement	Spécifie l'orientation du texte de l'étiquette par rapport à une icône.

Evénements de la classe RadioButton

Le tableau suivant présente l'événement de la classe RadioButton.

Evénement	Description
RadioButton.click	Déclenché lorsque vous survolez un bouton radio ou un groupe de boutons radio avec la souris.

Evénements hérités de la classe UIObject

Le tableau suivant répertorie les événements de la classe RadioButton hérités de la classe UIObject.

Evénement	Description
UIObject.draw	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
UIObject.hide	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
UIObject.load	Diffusé lorsque des sous-objets sont créés.
UIObject.move	Diffusé lorsque l'objet a été déplacé.
UIObject.resize	Diffusé lorsqu'un objet a été redimensionné.
UIObject.reveal	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
UIObject.unload	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant répertorie les événements de la classe RadioButton hérités de la classe UIComponent.

Evénement	Description
`UIComponent.focusIn`	Diffusé lorsqu'un objet reçoit le focus.
`UIComponent.focusOut`	Diffusé lorsqu'un objet perd le focus.
`UIComponent.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIComponent.keyUp`	Diffusé lorsqu'une touche est relâchée.

Evénements hérités de la classe SimpleButton

Le tableau suivant présente l'événement de la classe RadioButton hérité de la classe SimpleButton.

Evénement	Description
`SimpleButton.click`	Diffusé lorsque le bouton de la souris est relâché sur un bouton ou si ce dernier a le focus et que la barre d'espace est enfoncée.

RadioButton.click

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.click = function(eventObj:Object) {
  // ...
};
radioButtonGroup.addEventListener("click", listenerObject);
```

Utilisation 2 :

```
on (click) {
  // ...
}
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque l'utilisateur clique à l'aide de la souris (bouton de la souris enfoncé et relâché) sur le bouton radio ou si le bouton radio est sélectionné en utilisant les boutons fléchés. L'événement est également diffusé si vous appuyez sur la barre d'espace ou sur les boutons fléchés lorsqu'un groupe de boutons radio a le focus mais qu'aucun des boutons radio du groupe n'est sélectionné.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence de composant (*radioButtonInstance*) distribue un événement (ici, `click`) qui est géré par une fonction (également appelée *gestionnaire*), sur un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée au déclenchement de l'événement. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. L'objet événement est doté de propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence RadioButton. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence du composant. Par exemple, le code suivant, associé au bouton radio `myRadioButton`, envoie « _level0.myRadioButton » vers le panneau Sortie :

```
on (click) {
  trace(this);
}
```

Exemple

Cet exemple crée trois boutons radio, les positionne sur la scène et crée un écouteur pour l'événement click. Lorsqu'un utilisateur clique sur l'un des trois boutons radio, l'écouteur affiche le nom de l'occurrence du bouton radio sélectionné.

Vous devez d'abord faire glisser un composant RadioButton depuis le panneau Composants vers la bibliothèque du document en cours, puis ajouter le code suivant à l'image 1 du scénario principal :

```
/**
 Requiert :
  - Composant RadioButton dans la bibliothèque
*/
```

```
import mx.controls.RadioButton;

this.createClassObject(RadioButton, "first_rb", 10, {label:"first",
  groupName:"radioGroup"});
this.createClassObject(RadioButton, "second_rb", 20, {label:"second",
  groupName:"radioGroup"});
this.createClassObject(RadioButton, "third_rb", 30, {label:"third",
  groupName:"radioGroup"});

// Positionnement des boutons radio sur la scène.
second_rb.move(0, first_rb.y + first_rb.height);
third_rb.move(0, second_rb.y + second_rb.height);

// Création d'un objet écouteur.
var rbListener:Object = new Object();
rbListener.click = function(evt_obj:Object){
   trace("The selected radio instance is " + evt_obj.target.selection);
}
// Ajout de l'écouteur.
radioGroup.addEventListener("click", rbListener);
```

RadioButton.data

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

radioButtonInstance.data

Description

Propriété : spécifie les données à associer à une occurrence de RadioButton. La définition de cette propriété annule la valeur du paramètre des données défini lors de la programmation. La propriété `data` peut être de n'importe quel type de données.

Exemple

L'exemple suivant affecte la valeur de données `0xFF00FF` et l'étiquette `#FF00FF` à l'occurrence RadioButton `my_rb`. Il crée ensuite un écouteur pour un événement click et affiche la valeur de données du bouton lorsqu'un utilisateur clique sur le bouton.

Vous devez d'abord faire glisser un composant RadioButton depuis le panneau Composants jusqu'à la bibliothèque du document actif, puis ajouter le code suivant à l'image 1 du scénario principal :

```
/**
 Requiert :
  - Composant RadioButton dans la bibliothèque
*/

this.createClassObject(mx.controls.RadioButton, "my_rb", 10,
  {label:"first", groupName:"radioGroup"});

my_rb.data = 0xFF00FF;
my_rb.label = "#FF00FF";

var rbListener:Object = new Object();
rbListener.click = function(evt_obj:Object){
   trace("The data value for my_rb is " + my_rb.data);
}
// Ajout de l'écouteur.
my_rb.addEventListener("click", rbListener);
```

RadioButton.groupName

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

radioButtonInstance.groupName

radioButtonGroup.groupName

Description

Propriété : définit le nom du groupe d'une occurrence de bouton radio ou d'un groupe de boutons radio. Vous pouvez utiliser cette propriété afin d'obtenir ou de définir un nom de groupe pour une occurrence de bouton radio ou pour un groupe de boutons radio. L'appel de cette méthode annule la valeur du paramètre groupName définie lors de la programmation. La valeur par défaut est « `radioGroup` ».

Exemple

L'exemple suivant définit le nom d'un groupe de trois boutons radio sur `myrbGroup`. Il positionne les boutons puis crée un écouteur pour un événement click sur le groupe de boutons radio. Lorsque l'utilisateur clique sur un bouton radio, l'exemple affiche la propriété `groupName` pour le bouton sur lequel vous avez cliqué.

Vous devez d'abord faire glisser un composant RadioButton depuis le panneau Composants jusqu'à la bibliothèque du document actif, puis ajouter le code suivant à l'image 1 du scénario principal :

```
/**
 Requiert :
  - Composant RadioButton dans la bibliothèque
*/

import mx.controls.RadioButton;

this.createClassObject(RadioButton, "first_rb", 10, {label:"first",
  groupName:"myrbGroup"});
this.createClassObject(RadioButton, "second_rb", 20, {label:"second",
  groupName:"myrbGroup"});
this.createClassObject(RadioButton, "third_rb", 30, {label:"third",
  groupName:"myrbGroup"});

// Positionnement des boutons radio sur la scène.
second_rb.move(0, first_rb.y + first_rb.height);
third_rb.move(0, second_rb.y + second_rb.height);

// Création d'un objet écouteur.
var rbListener:Object = new Object();
rbListener.click = function(evt_obj:Object){
   trace("The selected radio button group name is " +
  evt_obj.target.groupName);
}
// Ajout de l'écouteur.
myrbGroup.addEventListener("click", rbListener);
```

RadioButton.label

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

radioButtonInstance.label

Description

Propriété : spécifie le texte de l'étiquette du bouton radio. Par défaut, l'étiquette apparaît à droite du bouton radio. L'appel de cette méthode annule le paramètre label spécifié lors de la programmation. Si le texte de l'étiquette est trop long et ne rentre pas dans le cadre de sélection du composant, il est rogné. Vous pouvez appeler `RadioButton.setSize()` pour augmenter la taille de la zone d'étiquette, mais le texte ne passe pas à la ligne suivante.

Pour fournir une étiquette avec du texte qui englobe, vous pouvez combiner un composant RadioButton sans étiquette et un composant TextArea agissant comme composant RadioButton unique avec du texte enveloppant. L'exemple suivant crée un bouton radio de ce type. Il suppose que la bibliothèque contient un composant RadioButton et un composant TextArea et désactive la bordure pour le composant TextArea. La propriété `label` serait undefined dans ce cas, si vous y avez accédé.

```
this.createClassObject(mx.controls.RadioButton, "sameas_rb", 1,
  {groupName:"myGroup"});
sameas_rb.move(0,30)
this.createClassObject(mx.controls.TextArea, "message_ta", 2);
message_ta.setSize(200, 60);
// Désactivation de la bordure pour le composant TextArea.
message_ta.borderStyle = "none";
message_ta.wordWrap = true;
message_ta.text = "Click here if your shipping information is the same as
  your billing information.";
message_ta.move(20, 30);
```

Exemple

L'exemple suivant crée un bouton radio et lui affecte une étiquette « Remove from list. »

Vous devez d'abord ajouter un composant RadioButton depuis le panneau Composants vers la bibliothèque du document en cours, puis ajouter le code suivant à l'image 1 du scénario principal.

```
/**
 Requiert :
  - Composant RadioButton dans la bibliothèque
*/

this.createClassObject(mx.controls.RadioButton, "my_rb", 10);

// Redimensionnement du composant RadioButton.
my_rb.setSize(200, my_rb.height);
my_rb.label = "Remove from list";
```

RadioButton.labelPlacement

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

radioButtonInstance.labelPlacement

radioButtonGroup.labelPlacement

Description

Propriété : chaîne indiquant la position de l'étiquette par rapport à un bouton radio.
Pour pouvez définir cette propriété pour une occurrence individuelle ou pour un groupe de boutons radio. Si vous définissez la propriété pour un groupe, l'étiquette est placée à l'endroit approprié pour chaque bouton radio du groupe.

Quatre valeurs sont possibles :

- « `right` » Le bouton radio est placé dans le coin supérieur gauche du cadre de délimitation. L'étiquette est placée à droite du bouton radio.
- « `left` » Le bouton radio est placé dans le coin supérieur droit du cadre de délimitation. L'étiquette est placée à gauche du bouton radio.

- « `bottom` » L'étiquette est placée sous le bouton radio. Le bouton radio et l'étiquette sont centrés horizontalement et verticalement. Si la dimension du cadre de sélection du bouton radio est trop réduite, l'étiquette est rognée.
- « `top` » L'étiquette est placée au-dessus du bouton radio. Le bouton radio et l'étiquette sont centrés horizontalement et verticalement. Si la dimension du cadre de sélection du bouton radio est trop réduite, l'étiquette est rognée.

Exemple

Le code suivant crée trois boutons radio et utilise la propriété `labelPlacement` pour placer l'étiquette pour le deuxième bouton à gauche du bouton.

Vous devez d'abord faire glisser un composant RadioButton depuis le panneau Composants vers la bibliothèque du document en cours, puis ajouter le code suivant à l'image 1 du scénario principal :

```
/**
 Requiert :
  - Composant RadioButton dans la bibliothèque
*/

import mx.controls.RadioButton;

this.createClassObject(RadioButton, "first_rb", 10, {label:"first",
  groupName:"radioGroup"});
this.createClassObject(RadioButton, "second_rb", 20, {label:"second",
  groupName:"radioGroup"});
this.createClassObject(RadioButton, "third_rb", 30, {label:"third",
  groupName:"radioGroup"});

// Positionnement des boutons radio sur la scène.
second_rb.move(0, first_rb.y + first_rb.height);
third_rb.move(0, second_rb.y + second_rb.height);

second_rb.labelPlacement = "left";
```

RadioButton.selected

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

radioButtonInstance.selected

radioButtonGroup.selected

Description

Propriété : valeur booléenne qui définit l'état du bouton radio sur sélectionné (`true`) et désélectionne le bouton radio sélectionné précédemment ou définit l'état du bouton radio sur désélectionné (`false`).

Exemple

L'exemple suivant crée trois boutons radio dans un groupe de boutons radio, les positionne et définit la propriété `selected` sur `true` pour la placer dans l'état sélectionné.

Vous devez d'abord faire glisser un composant RadioButton depuis le panneau Composants jusqu'à la bibliothèque du document actif, puis ajouter le code suivant à l'image 1 du scénario principal :

```
/**
 Requiert :
  - Composant RadioButton dans la bibliothèque
*/

import mx.controls.RadioButton;

this.createClassObject(RadioButton, "first_rb", 10, {label:"first",
  groupName:"radioGroup"});
this.createClassObject(RadioButton, "second_rb", 20, {label:"second",
  groupName:"radioGroup"});
this.createClassObject(RadioButton, "third_rb", 30, {label:"third",
  groupName:"radioGroup"});

// Positionnement des boutons radio sur la scène.
second_rb.move(0, first_rb.y + first_rb.height);
third_rb.move(0, second_rb.y + second_rb.height);

first_rb.selected = true;
```

RadioButton.selectedData

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

radioButtonGroup.selectedData

Description

Propriété : sélectionne le bouton radio avec la valeur de données spécifiée et désélectionne le bouton radio sélectionné précédemment. Si la propriété `data` n'est pas spécifiée pour une occurrence sélectionnée, la valeur `label` de l'occurrence sélectionnée est sélectionnée et renvoyée. La propriété `selectedData` peut être de n'importe quel type de données.

Exemple

L'exemple suivant crée trois boutons radio dans un groupe de boutons radio, les positionne et sélectionne le bouton ayant une valeur de données de 10 (le deuxième bouton).

Vous devez d'abord faire glisser un composant RadioButton depuis le panneau Composants vers la bibliothèque du document en cours, puis ajouter le code suivant à l'image 1 du scénario principal :

```
/**
 Requiert :
  - Composant RadioButton dans la bibliothèque
*/

import mx.controls.RadioButton;

this.createClassObject(RadioButton, "first_rb", 10, {label:"first", data:5,
  groupName:"radioGroup"});
this.createClassObject(RadioButton, "second_rb", 20, {label:"second",
  data:10, groupName:"radioGroup"});
this.createClassObject(RadioButton, "third_rb", 30, {label:"third",
  data:15, groupName:"radioGroup"});

// Positionnement des boutons radio sur la scène.
second_rb.move(0, first_rb.y + first_rb.height);
third_rb.move(0, second_rb.y + second_rb.height);

radioGroup.selectedData = 10;
```

RadioButton.selection

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

radioButtonInstance.selection

radioButtonGroup.selection

Description

Propriété : se comporte de manière différente si vous obtenez ou définissez la propriété. Si vous obtenez la propriété, elle renvoie la référence objet du bouton radio actuellement sélectionné dans un groupe de boutons radio. Si vous définissez la propriété, elle sélectionne le bouton radio spécifié (transmis comme référence objet) dans un groupe de boutons radio et désélectionne le bouton radio précédemment sélectionné.

Exemple

L'exemple suivant crée trois boutons radio dans un groupe de boutons radio, les positionne et crée un écouteur pour un événement click sur le groupe de boutons radio. Lorsque l'utilisateur clique sur un bouton radio, l'écouteur utilise la propriété `selection` pour afficher le nom de l'occurrence du bouton sur lequel vous avez cliqué.

Vous devez d'abord faire glisser un composant RadioButton depuis le panneau Composants vers la bibliothèque du document en cours, puis ajouter le code suivant à l'image 1 du scénario principal :

```
/**
 Requiert :
  - Composant RadioButton dans la bibliothèque
*/

import mx.controls.RadioButton;

this.createClassObject(RadioButton, "first_rb", 10, {label:"first",
  groupName:"radioGroup"});
this.createClassObject(RadioButton, "second_rb", 20, {label:"second",
  groupName:"radioGroup"});
this.createClassObject(RadioButton, "third_rb", 30, {label:"third",
  groupName:"radioGroup"});

// Positionnement des boutons radio sur la scène.
```

```
second_rb.move(0, first_rb.y + first_rb.height);
third_rb.move(0, second_rb.y + second_rb.height);

// Création d'un objet écouteur.
var rbListener:Object = new Object();
rbListener.click = function(evt_obj:Object){
 trace("The selected radio instance is " + radioGroup.selection);
}
// Ajout de l'écouteur.//
radioGroup.addEventListener("click", rbListener);
```

CHAPITRE 36

Composant RadioButtonGroup

36

Pour plus d'informations sur la classe RadioButtonGroup, reportez-vous à Composant RadioButton.

CHAPITRE 37

Composant RDBMSResolver

37

Les composants Resolver s'utilisent en association avec le composant DataSet (partie de la fonctionnalité de gestion des données dans l'architecture de données Flash) pour enregistrer des changements dans une source de données externe. Ils comprennent le composant RDBMSResolver et le composant XUpdateResolver. Ils prennent un paquet delta (renvoyé par `DataSet.deltaPacket`) et le convertissent en un paquet de mise à jour dans un format approprié au type de resolver. Le paquet de mise à jour peut ensuite être transmis à la source de données externe par l'un des composants Connector. Les composants Resolver ne sont pas visibles lors de l'exécution.

REMARQUE

Le composant RDBMSResolver est pris en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Le composant RDBMSResolver crée un paquet de mise à jour XML qui peut facilement être analysé en instructions SQL pour mettre à jour une base de données relationnelle.
Le composant RDBMSResolver est connecté à la propriété `DeltaPacket` d'un composant DataSet ; il envoie son propre paquet de mise à jour à un connecteur, qui lui renvoie les erreurs serveur ; le composant Resolver transmet ces erreurs au composant DataSet. Tous ces processus utilisent les propriétés de liaison.

Pour plus d'informations sur l'architecture de données Flash, reportez-vous à *Résolution des données* dans *Utilisation de Flash*. Pour plus d'informations sur les bases de données relationnelles, reportez-vous à *Résolution des données pour une base de données relationnelle* dans *Utilisation de Flash*. Pour un exemple complet d'une application mettant à jour des données à l'aide du composant RDBMSResolver, visitez le site Adobe à l'adresse suivante www.adobe.com/devnet/mx/flash/articles/delta_packet.html.

CONSEIL	Vous pouvez utiliser le composant RDBMSResolver pour envoyer des mises à jour de données vers un objet que vous écrivez qui peut analyser des données XML et générer des instructions SQL par rapport une base de données (par exemple, une page ASP, un servlet Java ou un composant ColdFusion).

Utilisation du composant RDBMSResolver

Vous utilisez le composant RDBMSResolver uniquement lorsque votre application Flash contient un composant DataSet et doit renvoyer une mise à jour à la source de données. Ce composant traduit les données que vous souhaitez renvoyer à une base de données relationnelles.

Paramètres de RDBMSResolver

Vous pouvez définir les paramètres de programmation suivants pour chaque occurrence de RDBMSResolver en utilisant l'onglet Paramètres de l'inspecteur des composants :

TableName est une chaîne représentant **le nom (dans le XML) de** la table de base de données à mettre à jour. Cette chaîne doit correspondre au nom de l'élément `RDBMSResolver.fieldInfo` à mettre à jour. S'il n'existe aucune mise à jour de ce champ, ce paramètre doit être vide (valeur par défaut).

UpdateMode est un énumérateur qui détermine la manière dont les champs-clés sont identifiés au moment de la génération du paquet de mise à jour XML. Les valeurs possibles sont :

- `umUsingAll` Utilise les anciennes valeurs de tous les champs modifiés pour identifier l'enregistrement à mettre à jour. Il s'agit de la valeur la plus sûre pour la mise à jour car elle garantit que l'enregistrement n'a pas été modifié par un autre utilisateur depuis que vous l'avez récupéré. Toutefois, elle prend du temps et génère un paquet de mise à jour plus volumineux.

- `umUsingModified` Utilise les anciennes valeurs de tous les champs modifiés pour identifier l'enregistrement à mettre à jour. Cette valeur garantit qu'aucun autre utilisateur n'a modifié les mêmes champs de l'enregistrement depuis que vous l'avez récupéré.
- `umUsingKey` La valeur par défaut. Ce paramètre utilise l'ancienne valeur des champs-clés. Il implique un « modèle de contrôle des accès concurrents optimiste », utilisé aujourd'hui par la plupart des systèmes de base de données, et garantit que vous modifiez l'enregistrement que vous avez récupéré de la base de données. Vos modifications remplacent celles qu'un autre utilisateur a apportées aux mêmes données.

NullValue est une chaîne représentant une valeur de champ null. Vous pouvez personnaliser ce paramètre pour éviter de la confondre avec une chaîne vide (`""`) ou une autre valeur valide. La valeur par défaut est `{_NULL_}`.

FieldInfo est une collection représentant un ou plusieurs champs-clés qui identifient les enregistrements de façon unique. Si votre source de données est une table de base de données, cette dernière doit posséder un ou plusieurs champs qui identifient ses enregistrements de façon unique. En outre, certains champs peuvent avoir été calculés ou joints à partir d'autres tables. Ces champs doivent être identifiés pour que les champs-clés puissent être définis dans le paquet de mise à jour XML et pour que les champs qui ne doivent pas être mis à jour soient omis du paquet de mise à jour XML.

Le paramètre FieldInfo vous permet d'utiliser des propriétés identifiant les champs qui nécessitent un traitement particulier. Chaque élément de la collection contient trois propriétés :

- `FieldName` Nom d'un champ. Il doit correspondre à un nom de champ du composant DataSet.
- `OwnerName` Valeur facultative utilisée pour identifier des champs qui ne sont pas inclus dans la table définie dans le paramètre TableName du composant RDBMSResolver. Si cette propriété possède la même valeur que celle du paramètre TableName ou si elle est vide, le champ est généralement inclus dans le paquet de mise à jour XML. Si vous entrez une autre valeur, le champ est exclu du paquet de mise à jour.
- `IsKey` Propriété booléenne que vous devez définir sur `true` pour que tous les champs-clés de la table soient mis à jour.

L'exemple suivant montre les éléments FieldInfo créés pour mettre à jour les champs de la table client. Vous devez identifier les champs-clés de la table client. La table client possède un champ-clé unique, `id` ; vous devez donc créer un élément de champ avec les valeurs suivantes :

```
FieldName = "id"
OwnerName = <--! leave this value blank -->
IsKey = "true"
```

Le champ `custType` est également ajouté à l'aide d'une jonction dans la requête. Etant donné que ce champ doit être exclu de la mise à jour, vous créez donc un élément de champ avec les valeurs suivantes :

```
FieldName = "custType"
OwnerName = "JoinedField"
IsKey = "false"
```

Une fois les éléments de champ définis, Flash Player peut les utiliser pour générer automatiquement le paquet de mise à jour XML qui servira à mettre à jour une table.

REMARQUE

Le paramètre fieldInfo utilise une fonction de Flash appelée Editeur de collecte. Lorsque vous sélectionnez le paramètre FieldInfo, vous pouvez utiliser la boîte de dialogue Editeur de collecte pour ajouter de nouveaux éléments FieldInfo et définir leurs propriétés `fieldName`, `ownerName` et `isKey` à partir d'un emplacement.

Flux de travaux courant pour le composant RDBMSResolver

Les étapes suivantes décrivent le flux de travaux classique pour le composant RDBMSResolver.

Pour utiliser un composant RDBMSResolver :

1. Vérifiez que vos paramètres de publication sont bien définis sur ActionScript 2.0.
2. Ajoutez deux occurrences du composant WebServiceConnector, une occurrence du composant DataSet et une occurrence du composant RDBMSResolver à votre application et donnez-leur des noms d'occurrence.

3. Sélectionnez le premier composant WebServiceConnector. Utilisez ensuite l'onglet Paramètres de l'inspecteur des composants pour entrer l'URL WSDL (Web Service Definition Language) pour un service Web qui expose des données d'un source de données externe.

REMARQUE	Le service Web doit renvoyer un tableau d'enregistrements à lier au jeu de données.

4. Utilisez l'onglet Liaisons de l'inspecteur de composants pour lier la première propriété results du composant WebServiceConnector à la propriété `dataProvider` du composant DataSet.
5. Sélectionnez le composant DataSet et utilisez l'onglet Liaisons de l'inspecteur des composants pour lier les éléments de données (champs DataSet) aux composants visuels dans votre application.
6. Liez la propriété `deltaPacket` du composant DataSet à la propriété `deltaPacket` du composant RDBMSResolver.

   Les instructions sur la mise à jour sont envoyées à partir du composant DataSet vers le composant RDBMSResolver lorsque la méthode `DataSet.applyUpdates()` est appelée.
7. Liez la propriété `updatePacket` du composant RDBMSResolver à la deuxième propriété `params` du composant WebServiceConnector pour renvoyer des données à une méthode qui analyse le paquet de mise à jour XML. Définissez le type de la propriété `params` sur auto-trigger de façon à ce que le connecteur envoie le paquet de mise à jour dès que la liaison des données a effectué la copie.
8. Ajoutez un déclencheur pour lancer l'opération de liaison des données : utilisez le comportement Déclencher une source de données associé à un bouton ou ajoutez des instructions ActionScript.

Outre ces étapes, vous pouvez également utiliser le composant RDBMSResolver pour créer des liaisons afin d'appliquer le paquet de résultats renvoyé du serveur au jeu de données.

Pour un exemple détaillé qui résout des données dans une base de données relationnelles à l'aide du composant RDBMSResolver, consultez les didacticiels sur DevNet à l'adresse suivante www.adobe.com/devnet/mx/flash/ data_integration.html.

Classe RDBMSResolver

Héritage MovieClip > RDBMSResolver

Nom du logiciel ActionScript mx.data.components.RDBMSResolver

Les méthodes, propriétés et événements de la classe RDBMSResolver vous permettent d'effectuer une connexion à un composant DataSet et de modifier des sources de données externes.

Méthodes du composant RDBMSResolver

Le tableau suivant indique la méthode de la classe RDBMSResolver.

Méthode	Description
`RDBMSResolver.addFieldInfo()`	Ajoute un nouvel élément à la collection `fieldInfo` qui est utilisé pour la configuration d'un composant RDBMSResolver de façon dynamique lors de l'exécution.

Propriétés du composant RDBMSResolver

Le tableau suivant répertorie les propriétés de la classe RDBMSResolver.

Propriété	Description
`RDBMSResolver.deltaPacket`	La propriété `deltaPacket` de l'objet DataSet doit être liée à cette propriété de façon à ce que la liaison la copie lors de l'appel à `DataSet.applyUpdates()` et que le composant Resolver crée le paquet de mise à jour.
`RDBMSResolver.fieldInfo`	Ensemble de champs avec des propriétés qui identifient les champs DataSet nécessitant un traitement particulier, soit parce qu'il s'agit de champs-clés, soit parce qu'ils ne peuvent pas être mis à jour.
`RDBMSResolver.nullValue`	Chaîne placée dans le paquet de mise à jour pour indiquer que la valeur d'un champ est `null`.
`RDBMSResolver.tableName`	Identifie la table de base de données à mettre à jour.
`RDBMSResolver.updateMode`	Valeurs déterminant la manière dont les champs-clés sont identifiés lorsque le paquet de mise à jour XML est généré.

Propriété	Description
RDBMSResolver.updatePacket	Paquet XML produit par ce composant Resolver qui contient les changements du paquet delta du jeu de données.
RDBMSResolver.updateResults	Paquet delta contenant les résultats d'une mise à jour renvoyée par le serveur via un connecteur.

Evénements du composant RDBMSResolver

Le tableau suivant répertorie les événements de la classe RDBMSResolver.

Evénement	Description
RDBMSResolver.beforeApplyUpdates	Défini dans votre application ; appelé par le composant RDBMSResolver pour apporter des modifications personnalisées au paquet XML de la propriété updatePacket avant qu'il soit lié au connecteur.
RDBMSResolver.reconcileResults	Défini dans votre application ; appelé par le composant RDBMSResolver pour comparer deux paquets une fois que les résultats ont été reçus du serveur et appliqués au paquet delta.
RDBMSResolver.reconcileUpdates	Défini dans votre application ; appelé par le composant RDBMSResolver lorsque les résultats ont été reçus du serveur une fois que les mises à jour du paquet delta ont été appliquées.

RDBMSResolver.addFieldInfo()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
resolveData.addFieldInfo("fieldName", "ownerName", "isKey")
```

Paramètres

fieldName Chaîne qui indique le nom du champ décrit par cet objet d'informations.

ownerName Chaîne qui indique le nom de la table à laquelle ce champ appartient. Si ce nom est identique à la propriété `tableName` de l'occurrence RDBMSResolver, vous pouvez laisser ce paramètre vide (`""`).

isKey Valeur booléenne indiquant s'il s'agit d'un champ-clé.

Renvoie

Aucune.

Description

Méthode : ajoute un nouvel élément à la collection `fieldInfo` XML dans le paquet de mise à jour. Utilisez cette méthode si vous devez configurer un composant RDBMSResolver de façon dynamique lors de l'exécution plutôt que d'utiliser l'inspecteur de composants dans l'environnement de programmation.

Exemple

L'exemple suivant crée un composant RDBMSResolver et fournit le nom de la table, le nom du champ-clé, et empêche la mise à jour du champ `personTypeName` :

```
var myResolver:RDBMSResolver = new RDBMSResolver();
myResolver.tableName = "Customers";
// Définition du champ id en tant que champ-clé.
// et du champ personTypeName pour qu'il ne soit pas mis à jour.
myResolver.addFieldInfo("id", "", true);
myResolver.addFieldInfo("personTypeName", "JoinedField", false);
// Définition des liaisons de données.
//...
```

RDBMSResolver.beforeApplyUpdates

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

resolveData.beforeApplyUpdates(*eventObject*)

Paramètres

eventObject Objet événement Resolver : décrit les personnalisations apportées au paquet XML avant l'envoi de la mise à jour à la base de données par le biais du connecteur. Cet objet événement doit contenir les propriétés suivantes :

Propriété	Description
target	Objet : le composant Resolver produisant cet événement.
type	Chaîne ; nom de l'événement.
updatePacket	Objet XML : l'objet XML qui sera appliqué.

Renvoie

Aucune.

Description

Propriété : propriété de type deltaPacket. Elle reçoit un paquet delta à traduire en paquet de mise à jour et génère un paquet delta des résultats du serveur placés dans la propriété updateResults. Ce gestionnaire d'événement vous permet d'apporter des modifications personnalisées au paquet XML avant l'envoi des données mises à jour au connecteur.

Les messages dans la propriété updateResults sont considérés comme des erreurs. Cela signifie qu'un paquet delta contenant des messages est de nouveau ajouté au paquet delta pour être renvoyé lors de la prochaine transmission de ce paquet au serveur. Vous devez rédiger le code qui gère les deltas contenant des messages afin que ces derniers soient présentés à l'utilisateur et modifiés avant d'être ajoutés au paquet delta suivant.

Exemple

L'exemple suivant ajoute les données d'authentification de l'utilisateur au paquet XML :

```
on (beforeApplyUpdates) {
   // Ajout des données d'authentification de l'utilisateur.
   var userInfo = new XML("" + getUserId() + ""+getPassword() + "");
   updatePacket.firstChild.appendChild(userInfo);
}
```

RDBMSResolver.deltaPacket

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

resolveData.deltaPacket

Description

Propriété : propriété de type `deltaPacket`. Elle reçoit un paquet delta à traduire en paquet de mise à jour et génère un paquet delta des résultats du serveur placés dans la propriété `updateResults`.

Les messages dans la propriété `updateResults` sont considérés comme des erreurs. Cela signifie qu'un paquet delta contenant des messages est de nouveau ajouté au paquet delta pour être renvoyé lors de la prochaine transmission de ce paquet au serveur. Vous devez rédiger le code qui gère les deltas contenant des messages afin que ces derniers soient présentés à l'utilisateur et modifiés avant d'être ajoutés au paquet delta suivant.

RDBMSResolver.fieldInfo

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

resolveData.fieldInfo

Description

Propriété ; spécifie la collection d'un nombre illimité de champs, avec des propriétés qui identifient les champs DataSet nécessitant un traitement particulier, soit parce qu'il s'agit de champs-clés, soit parce qu'ils ne peuvent pas être mis à jour (pour obtenir des informations supplémentaires sur l'ajout d'un champ, reportez-vous à RDBMSResolver.addFieldInfo()). Chaque élément fieldInfo de la collection contient trois propriétés :

Propriété	Description
fieldName	Nom du champ de cas particulier. Le nom de ce champ doit correspondre à un nom de champ du composant DataSet.
ownerName	Propriété facultative. Si ce champ n'« appartient » pas à la table définie dans la propriété RDBMSResolver.tableName, OwnerName est le nom du propriétaire de ce champ. Si OwnerName possède la même valeur que RDBMSResolver.tableName ou qu'elle est vide, le champ est généralement inclus dans le paquet de mise à jour XML. Si OwnerName n'a aucune de ces valeurs, ce champ est exclu du paquet de mise à jour.
isKey	Valeur booléenne : si elle est définie sur true, tous les champs-clés de la table sont mis à jour.

RDBMSResolver.nullValue

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

resolveData.nullValue

Description

Propriété ; chaîne utilisée pour affecter une valeur null à un champ. Vous pouvez personnaliser cette propriété pour éviter de la confondre avec une chaîne vide ("") ou une autre valeur valide. La chaîne par défaut est {_NULL_}.

RDBMSResolver.reconcileResults

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

resolveData.reconcileResults(*eventObject*)

Paramètres

eventObject Objet événement Resolver : décrit l'objet événement utilisé pour comparer deux paquets de mise à jour. Cet objet événement doit contenir les propriétés suivantes :

Propriété	Description
target	Objet : composant Resolver diffusant cet événement.
type	Chaîne ; nom de l'événement.

Renvoie

Aucune.

Description

Evénement : diffusé par le composant RDBMSResolver pour comparer deux paquets une fois que les résultats ont été reçus du serveur et appliqués au paquet delta.

Un paquet `updateResults` peut contenir les résultats des opérations spécifiées dans le paquet delta et des informations relatives aux mises à jour effectuées par d'autres clients. Lorsqu'un nouveau paquet de mise à jour est reçu, les résultats des opérations et les mises à jour de la base de données sont séparés en deux paquets de mise à jour et placés séparément dans la propriété `deltaPacket`. L'événement `reconcileResults` est diffusé juste avant que le paquet delta contenant les résultats des opérations soit envoyé à l'aide de la liaison des données.

Exemple

L'exemple suivant reconstitue deux paquets de mise à jour, puis renvoie et efface les mises à jour en cas de succès :

```
on (reconcileResults) {
  // Examen des résultats.
  if (examine(updateResults)) {
    myDataSet.purgeUpdates();
  } else {
    displayErrors(results);
  }
}
```

RDBMSResolver.reconcileUpdates

Disponibilité

Flash Player 7.

Edition

Flash Professional 8.

Utilisation

`resolveData``.reconcileUpdates(`*`eventObject`*`)`

Paramètres

`eventObject` Objet événement Resolver : décrit les personnalisations apportées au paquet XML avant l'envoi de la mise à jour à la base de données par le biais du connecteur. Cet objet événement doit contenir les propriétés suivantes :

Propriété	Description
`target`	Objet ; composant Resolver diffusant cet événement.
`type`	Chaîne ; nom de l'événement.

Renvoie

Aucun.

Description

Evénement : diffusé par le composant RDBMSResolver lorsque les résultats ont été reçus du serveur une fois que les mises à jour du paquet delta ont été appliquées. Un paquet `updateResults` peut contenir à la fois les résultats des opérations spécifiées dans le paquet delta et des informations relatives aux mises à jour effectuées par d'autres clients. Lorsqu'un nouveau paquet de mise à jour est reçu, les résultats des opérations et les mises à jour de la base de données sont séparés en deux paquets delta placés séparément dans la propriété `deltaPacket`. L'événement `reconcileUpdates` est diffusé juste avant que le paquet delta contenant les mises à jour de la base données soit envoyé à l'aide de la liaison des données.

Exemple

L'exemple suivant reconstitue deux résultats et efface les mises à jour en cas de succès :

```
on (reconcileUpdates) {
  // Examen des résultats.
  if (examine(updateResults)) {
    myDataSet.purgeUpdates();
  } else {
    displayErrors(results);
  }
}
```

RDBMSResolver.tableName

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`resolveData``.tableName`

Description

Propriété : une chaîne représentant le nom de la table de base de données devant être mise à jour dans le paquet XML. Elle détermine également les champs à envoyer dans le paquet de mise à jour. A cette fin, le composant RDBMSResolver compare la valeur de cette propriété à la valeur fournie pour la propriété `fieldInfo.ownerName`. Si un champ ne comporte aucune entrée dans la propriété de collection `fieldInfo`, il est placé dans le paquet de mise à jour. S'il comporte une entrée dans la propriété de collection `fieldInfo` et que sa valeur `ownerName` est vide ou identique à la propriété `tableName` du composant RDBMSResolver, le champ est placé dans le paquet de mise à jour. S'il comporte une entrée dans la propriété de collection `fieldInfo` et que sa valeur `ownerName` n'est pas vide et différente de la propriété `tableName` du composant RDBMSResolver, le champ n'est pas placé dans le paquet de mise à jour.

RDBMSResolver.updateMode

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

`resolveData``.updateMode`

Description

Propriété : contient plusieurs valeurs déterminant la manière dont les champs-clés sont identifiés lorsque le paquet de mise à jour XML est généré. Les valeurs de cette propriété peuvent avoir les chaînes suivantes :

Valeur	Description
`"umUsingAll"`	Utilise les anciennes valeurs de tous les champs modifiés pour identifier l'enregistrement à mettre à jour. Il s'agit de la valeur la plus sûre pour la mise à jour, car elle garantit que l'enregistrement n'a pas été modifié par un autre utilisateur depuis que vous l'avez récupéré. Toutefois, elle nécessite davantage de temps et génère un paquet de mise à jour plus volumineux.

Valeur	Description
`"umUsingModified"`	Utilise les anciennes valeurs de tous les champs modifiés pour identifier l'enregistrement à mettre à jour. Cette valeur garantit qu'aucun autre utilisateur n'a modifié les mêmes champs de l'enregistrement depuis que vous l'avez récupéré.
`"umUsingKey"`	La valeur par défaut. Ce paramètre utilise l'ancienne valeur des champs-clés. Il implique un « modèle de contrôle des accès concurrents optimiste », utilisé aujourd'hui par la plupart des systèmes de base de données, et garantit que vous modifiez l'enregistrement que vous avez récupéré de la base de données. Vos modifications remplacent celles qu'un autre utilisateur a apportées aux mêmes données.

RDBMSResolver.updatePacket

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

resolveData.updatePacket

Description

Propriété : propriété de type XML contenant un paquet XML utilisée pour établir une liaison avec une propriété de connecteur qui renvoie le paquet de mise à jour contenant les changements traduits au serveur afin de permettre la mise à jour de la source de données. Il s'agit d'un document XML contenant le paquet des changements apportés au DataSet.

RDBMSResolver.updateResults

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

resolveData.updateResults

Description

Propriété : un paquet delta contenant les résultats d'une mise à jour renvoyée par le serveur via un connecteur. Utilisez cette propriété pour transmettre les erreurs et les données mises à jour à partir du serveur vers un jeu de données ; par exemple, lorsque le serveur affecte de nouveaux ID pour un champ auto-affecté. Liez cette propriété à la propriété `results` d'un connecteur afin que celui-ci puisse recevoir les résultats d'une mise à jour et retransmettre les résultats au jeu de données.

Les messages dans la propriété `updateResults` sont considérés comme des erreurs.
Cela signifie qu'un paquet delta contenant des messages est de nouveau ajouté au paquet delta pour être renvoyé lors de la prochaine transmission de ce paquet au serveur. Vous devez rédiger le code qui gère les deltas contenant des messages afin que ces derniers soient présentés à l'utilisateur et modifiés avant d'être ajoutés au paquet delta suivant.

CHAPITRE 38

Classe RectBorder

38

La classe RectBorder est utilisée comme bordure de la plupart des composants. Une implémentation séparée de cette classe est fournie par chaque thème qui possède son propre jeu de styles de bordure et de propriétés qu'il prend en charge.

REMARQUE

La classe RectBorder est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Vous interagissez principalement avec la classe RectBorder en définissant des styles sur d'autres composants. Par exemple, lorsque vous incluez un composant List dans un document et définissez la propriété de style `borderStyle`, le composant List crée une occurrence RectBorder qui utilise le paramètre `borderStyle` de la liste. Vous pouvez également créer une implémentation personnalisée de RectBorder pour appliquer une enveloppe à la bordure de tous les composants utilisant RectBorder.

La classe RectBorder possède quatre styles d'affichage standard : `none`, `inset`, `outset` et `solid`.

Le thème Halo ajoute également quatre styles d'affichage spéciaux utilisés par des composants spécifiques.

Style spécial	Composant qui l'utilise
`default`	Window
`alert`	Alert
`dropDown`	ComboBox et DateField
`menuBorder`	Menu et MenuBar

Le comportement et les propriétés de style RectBorder décrits ici sont cohérents pour tous les composants qui utilisent la classe RectBorder.

Utilisation des styles avec la classe RectBorder

Vous pouvez définir des propriétés de style afin de modifier l'aspect d'une occurrence de RectBorder. Une occurrence de RectBorder utilise les styles suivants :

- `borderCapColor`
- `borderColor`
- `buttonColor`
- `highlightColor`
- `shadowCapColor`
- `shadowColor`
- `themeColor`

Les styles disponibles sur une occurrence de RectBorder particulière dépendent du thème utilisé et du style de bordure défini sur le composant. Pour une démonstration interactive illustrant la relation entre le thème, le style de bordure et les propriétés de style de couleur disponibles, reportez-vous à *Utilisation des composants ActionScript 2.0* de l'Aide.

Les quatre styles Halo spéciaux—`default`, `alert`, `dropDown` et `menuBorder`—comportent certaines lignes dont les couleurs ne peuvent pas être définies au moyen des styles. Vous pouvez modifier ces couleurs uniquement en créant un thème personnalisé et en modifiant l'ActionScript approprié dans l'implémentation RectBorder personnalisée.

Pour définir un style de bordure à l'aide de setStyle :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser un composant TextArea sur la scène et nommez l'occurrence `my_ta`.
3. Dans la première image du scénario principal, ajoutez le code ActionScript suivant dans le panneau Actions :

```
my_ta.setStyle("borderStyle", "alert");
```

> **REMARQUE** Vous pouvez définir le borderStyle sur « alert », car vous utilisez le thème par défaut (Halo). Si vous utilisez un thème différent, les quatre styles Halo « spéciaux », y compris « alert », risquent de ne pas être disponibles.

4. Choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

Pour définir plusieurs styles de bordure comme paramètres de la méthode createClassObject :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Dans le panneau Actions, ajoutez le code ActionScript suivant à la première image du scénario principal :

   ```
   createClassObject(mx.controls.TextArea, "my_ta", 1, {borderStyle:
     "menuBorder", themeColor: "0x990000"});
   ```

   Pour plus d'informations, reportez-vous à `UIObject.createClassObject()`.
   Ou bien, si vous souhaitez définir plusieurs styles et les appliquer à plusieurs occurrences de composant, vous pouvez établir une nouvelle déclaration de style contenant les paramètres de style et l'associer aux occurrences de composant (voir « Définition de styles personnalisés pour des groupes de composants » dans *Utilisation des composants ActionScript 2.0*).
3. Choisissez Contrôle > Tester l'animation pour contrôler le fichier SWF.

Pour définir un style de bordure à l'aide du thème Sample :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser un composant Button sur la scène et nommez l'occurrence **my_btn**.

   Vous pouvez également créer l'occurrence en utilisant ActionScript, comme suit (faites bien d'abord glisser un composant Button vers la bibliothèque) :

   ```
   createClassObject(mx.controls.Button, "my_btn", 1);
   ```
3. Sélectionnez Fichier > Importer > Ouvrir une bibliothèque externe.
4. Ouvrez le fichier SampleTheme.fla qui se trouve dans :
   - Windows : \Program Files\Adobe\Flash CS3*langue*\Configuration\ComponentFLA\
   - Macintosh : HD/Applications/Adobe Flash CS3/Configuration/ComponentFLA/
5. Dans la bibliothèque SampleTheme.fla, recherchez le clip des éléments du composant Button dans Flash UI Components > Themes > MMDefault > Button Assets > Button Skin et faites-le glisser vers la bibliothèque de votre document actuel.

6. Dans le panneau Actions, ajoutez le code ActionScript suivant à la première image du scénario principal :

```
my_btn.setStyle("buttonColor", "0xFFFFFF");
my_btn.setStyle("borderStyle", "solid");
my_btn.setStyle("borderColor", "none");
```

REMARQUE	Si vous envisagez de définir plusieurs styles et que vous devez améliorer les performances du composant à l'exécution, vous pouvez définir une déclaration de style personnalisée contenant ces styles, puis l'associer à l'occurrence de composant (voir « Définition de styles personnalisés pour des groupes de composants » dans *Utilisation des composants ActionScript 2.0*).

Ou bien, vous pouvez ajouter ces paramètres à createClassObject, comme suit :

```
createClassObject(mx.controls.Button, "my_btn", 1, {buttonColor:
  "0xFFFFFF", borderStyle: "solid", borderColor: "none"});
```

7. Sélectionnez Contrôle > Tester l'animation pour contrôler le fichier SWF.

   Vous remarquerez que même lorsque « `borderColor` » est défini sur « `none` », le bouton présente une bordure grise. Dans ce cas, « `none` » ne signifie pas transparent, mais gris neutre.

Création d'une implémentation RectBorder personnalisée

La classe RectBorder est utilisée en tant qu'enveloppe de bordure dans la plupart des composants ActionScript 2.0. Les implémentations par défaut dans les thèmes Halo et Sample utilisent ActionScript pour dessiner la bordure. Une implémentation personnalisée doit utiliser ActionScript pour être enregistrée comme implémentation RectBorder et fournir une fonctionnalité de dimensionnement mais elle peut utiliser ActionScript ou des éléments graphiques pour les représentations visuelles.

Chaque implémentation RectBorder doit satisfaire les conditions suivantes :

- Elle doit étendre mx.skins.RectBorder ou l'une de ses sous-classes.
- Elle doit fournir une valeur de propriété `offset` ou implémenter la méthode `getBorderMetrics` pour renvoyer des informations de redimensionnement.
- Elle doit implémenter la méthode `drawBorder()` pour dessiner ou dimensionner la bordure.
- Elle doit prendre en charge les quatre styles standard ainsi que les quatre styles spéciaux.

  L'implémentation peut réutiliser des bordures standard pour des bordures spéciales, comme le thème Sample.
- Elle doit être enregistrée en tant qu'implémentation RectBorder.

Enregistrement globale de RectBorder

Tous les composants recherchent une référence à la classe RectBorder utilisée pour le document dans un emplacement centralisé, `_global.styles.rectBorderClass`.
Vous ne pouvez pas indiquer qu'un composant individuel doit utiliser une implémentation RectBorder différente. Pour personnaliser RectBorder pour un composant, vous devez utiliser la propriété de style `borderStyle`.

Exemple RectBorder personnalisé

Les implémentations RectBorder fournies par le thème Halo et le thème Sample utilisent l'API de dessin ActionScript pour dessiner les bordures de différents styles. L'exemple suivant démontre comment créer une implémentation RectBorder personnalisée qui utilise des symboles graphiques pour son affichage.

Pour créer une implémentation RectBorder personnalisée :

1. Créez un dossier dans le dossier Classes/mx/skins correspondant au nom du paquet personnalisé que vous utiliserez pour la bordure personnalisée.

   Pour cet exemple, utilisez `myTheme`.
2. Sélectionnez Fichier > Nouveau et choisissez Fichier ActionScript.
3. Enregistrez le fichier dans le même dossier que RectBorder.as.
4. Copiez le code ActionScript suivant dans le nouveau fichier :

```
import mx.core.ext.UIObjectExtensions;

class mx.skins.myTheme.RectBorder extends mx.skins.RectBorder
{
  static var symbolName:String = "RectBorder";
  static var symbolOwner:Object = RectBorder;
  var className:String = "RectBorder";

#include "../../core/ComponentVersion.as"

   // Toutes ces bordures ont des bords de même taille, 1 pixel.
  var offset:Number = 4;

  function init(Void):Void
  {
    super.init();
  }
```

```
    function drawBorder(Void):Void
    {
       // Les graphiques sont sur le scénario du symbole,
       // donc la seule chose que vous devez faire ici est
       // de dimensionner la bordure.
       _width = __width;
       _height = __height;
    }

    // Enregistrez la classe comme RectBorder pour tous les composants
    // à utiliser.
    static function classConstruct():Boolean
    {
      UIObjectExtensions.Extensions();
      _global.styles.rectBorderClass = RectBorder;
      _global.skinRegistry["RectBorder"] = true;
      return true;
    }
    static var classConstructed:Boolean = classConstruct();
    static var UIObjectExtensionsDependency = UIObjectExtensions;
}
```

Si vous n'utilisez pas le paquet `myTheme`, changez la déclaration de classe selon vos besoins.

5. Enregistrez le fichier.
6. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
7. Utilisez Insertion > Nouveau symbole pour créer un nouveau symbole de clip.
8. Nommez-le `RectBorder`.
9. Si les champs avancés ne sont pas affichés, cliquez sur Avancé.
10. Sélectionnez Exporter pour ActionScript

    L'identifiant est automatiquement renseigné avec `RectBorder`.
11. Définissez la classe sur le nom de classe complet de l'implémentation de bordure personnalisée.

    Cet exemple utilise `mx.skins.myTheme.RectBorder`.
12. Assurez-vous que l'option Exporter dans la première image est sélectionnée, puis cliquez sur OK.
13. Ouvrez le symbole RectBorder pour l'édition.
14. Dessinez les graphiques pour le symbole.

    Par exemple, dessinez un carré en filet sans remplissage. Pour rendre la bordure personnalisée facilement visible, définissez la couleur de la ligne sur le rouge vif.

15. Assurez-vous que les graphiques sont alignés contre le coin supérieur gauche avec les coordonnées *x* et *y* définies sur (0,0).

    Votre implémentation `drawBorder` personnalisée définit la largeur et la hauteur en fonction des exigences de composant.

16. Cliquez sur Précédent pour revenir au scénario principal.

17. Faites glisser plusieurs composants qui utilisent RectBorder sur la scène.

    Par exemple, faites glisser un composant List, TextArea et TextInput sur la scène.

18. Sélectionnez Contrôle > Tester l'animation.

Cet exemple crée une implémentation de bordure très simple avec des couleurs et des graphiques statiques. Il ne répond pas à des paramètres `borderStyle` différents ; il utilise toujours les mêmes graphiques, indépendamment de `borderStyle`. Pour des exemples d'implémentations de bordure plus complètes, consultez les exemples fournis pour les thèmes Halo et Sample.

CHAPITRE 39

Classe Screen

39

La classe Screen est la classe de base pour les écrans que vous créez dans le panneau Contour de l'écran d'Adobe Flash CS3 Professional. Les écrans représentent des conteneurs de haut niveau pour créer des applications et des présentations. Pour une présentation générale de l'utilisation des écrans, reportez-vous à *Utilisation des écrans* dans *Utilisation de Flash*.

Remarque : La fonction Ecrans est délaissée dans Flash CS3. Vous pouvez ouvrir et modifier les fichiers FLA composés d'écrans qui ont été créés à partir des précédentes versions de Flash, mais vous ne pouvez plus créer ce type de document. Par ailleurs, notez que la classe Screen est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript2.0 dans ses paramètres de publication.

La classe Screen contient deux sous-classes principales : Slide et Form.

La classe Slide fournit le comportement d'exécution des diaporamas. La classe Slide comporte des fonctions intégrées de navigation et de séquencement, ainsi que la capacité d'ajouter facilement des transitions entre les diapositives à l'aide des comportements. Les objets Slide conservent l'« état » et permettent à l'utilisateur d'atteindre la diapositive (ou l'état) suivante/précédente : lorsque la diapositive suivante apparaît, la précédente est masquée. Pour plus d'informations sur l'utilisation de la classe Slide pour contrôler les diaporamas, reportez-vous à « Classe Slide », à la page 1183.

La classe Form fournit l'environnement d'exécution des applications de formulaires. Les formulaires peuvent se superposer et contenir (ou être contenus par) d'autres composants. Contrairement aux diapositives, les formulaires ne disposent d'aucune capacité de séquencement ou de navigation. Pour plus d'informations, voir « Classe Form », à la page 761.

Les fonctionnalités de la classe Screen sont communes aux diapositives et aux formulaires.

Les écrans savent comment gérer leurs enfants Chaque écran comporte une propriété intégrée qui contient une liste d'écrans enfant de cet écran (collection). Cette collection est déterminée par la hiérarchie d'écrans dans le panneau Contour de l'écran. Les écrans peuvent posséder n'importe quel nombre d'enfants (y compris zéro) qui peuvent eux-mêmes avoir des enfants.

Les écrans peuvent masquer et afficher leurs enfants Etant donné qu'un écran est avant tout une collection de clips imbriqués, il peut contrôler la visibilité de ses enfants. Dans les applications de formulaires, tous les enfants d'un écran sont visibles en même temps par défaut ; dans les diaporamas, les écrans apparaissent généralement l'un après l'autre.

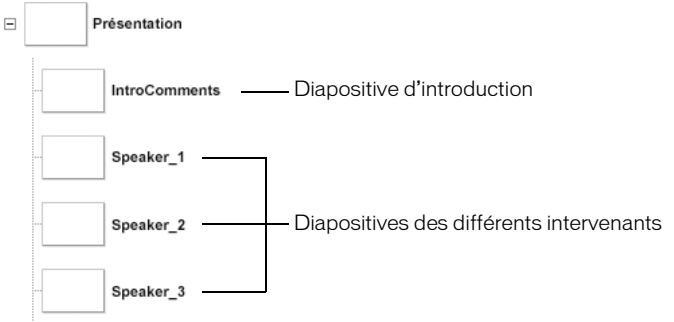
Les écrans diffusent des événements Vous pouvez par exemple déclencher la lecture d'un son ou d'une vidéo lorsqu'un écran spécifié apparaît.

Chargement de contenu externe dans des écrans

La classe Screen étend la classe Loader (voir « Composant Loader », à la page 845), ce qui facilite la gestion et le chargement de fichiers SWF et JPEG externes. La classe Loader contient une propriété `contentPath`, qui permet de spécifier l'URL d'un fichier SWF ou JPEG externe ou l'identifiant de liaison d'un clip de la bibliothèque.

Cette fonction vous permet de charger une arborescence d'écrans externe (ou tout fichier SWF externe) en tant qu'enfant de n'importe quel nœud d'écran. Ceci vous permet de rendre votre média composé d'écrans modulaire et de le séparer en fichiers SWF.

Prenons l'exemple d'un diaporama réalisé par trois intervenants, qui créent chacun leur propre partie. Vous pouvez demander à chaque intervenant de créer un diaporama distinct (fichier SWF). Il vous reste ensuite à créer un « diaporama maître » contenant trois balises d'emplacement, correspondant aux trois diaporamas créés par ces intervenants. Dans chaque balise d'emplacement, faites pointer la propriété `contentPath` vers l'un des fichiers SWF. Vous pouvez organiser le diaporama maître comme illustré ci-dessous :



Structure du diaporama du fichier « maître »

Supposons que les intervenants vous aient fourni trois fichiers SWF : speaker_1.swf, speaker_2.swf et speaker_3.swf. Vous pouvez facilement assembler le diaporama complet en définissant la propriété `contentPath` de chaque balise d'emplacement de diapositive, en utilisant soit l'inspecteur Propriétés, soit ActionScript, comme indiqué dans le code suivant :

```
Speaker_1.contentPath = speaker_1.swf;
Speaker_2.contentPath = speaker_2.swf;
Speaker_3.contentPath = speaker_3.swf;
```

Par défaut, lorsque vous définissez la propriété `contentPath` d'une diapositive pendant la création dans l'inspecteur Propriétés ou l'utilisation d'ActionScript (comme indiqué ci-dessus), le fichier SWF spécifié est chargé dès que le fichier SWF du « diaporama maître » est chargé. Pour réduire le temps de chargement initial, vous pouvez définir la propriété `contentPath` dans un gestionnaire `on(reveal)` associé à chaque diapositive.

```
// Associé à la diapositive Speaker_1.
on(reveal) {
  this.contentPath="speaker_1.swf";
}
```

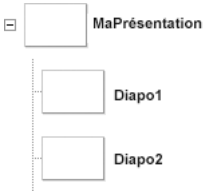
Vous pouvez également définir la propriété `autoLoad` de la diapositive sur `false`.
Vous pouvez ensuite appeler la méthode `load()` sur la diapositive lorsque cette dernière a été révélée. (La propriété `autoLoad` et la méthode `load()` sont héritées de la classe Loader.)

```
// Associé à la diapositive Speaker_1.
on(reveal) {
  this.load();
}
```

Référencement des écrans chargés avec ActionScript

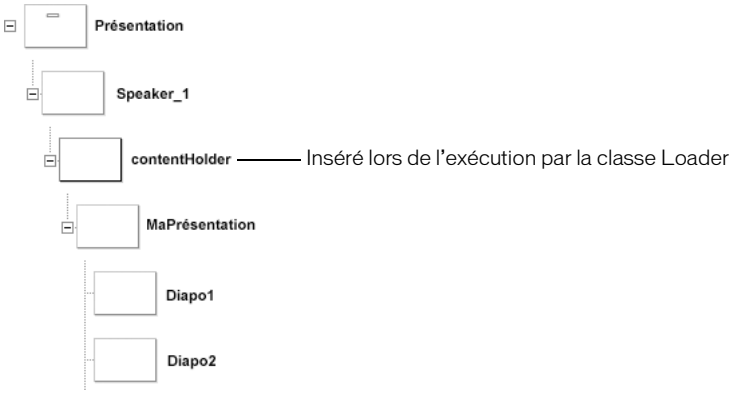
La classe Loader crée un clip interne nommé `contentNode` dans lequel elle charge le fichier SWF ou JPEG spécifié par la propriété `contentPath`. Cette animation ajoute un nœud d'écran supplémentaire dans le diaporama chargé, entre « la balise d'emplacement » de la diapositive (que vous avez créée dans le diaporama « maître », ci-dessus) et la première diapositive.

Imaginons par exemple que le fichier SWF créé pour la balise d'emplacement de la diapositive Speaker_1 (voir l'illustration ci-dessus) présente la structure suivante, comme indiqué dans le panneau Contour de l'écran :



Structure du diaporama du fichier SWF « Speaker 1 »

Lors de l'exécution, une fois le fichier SWF Speaker 1 chargé dans la balise d'emplacement de la diapositive, le diaporama complet présente la structure suivante :



Structure du diaporama « master » et « speaker » (à l'exécution)

Les propriétés et les méthodes des classes Screen, Slide et Form « ignorent » autant que possible le nœud `contentHolder`. En d'autres termes, la diapositive MyPresentation (et ses sous-diapositives) fait partie de l'arborescence de diapositives contiguës qui part de la diapositive Présentation. Elle n'est pas traitée comme une sous-arborescence séparée.

Classe Screen (API)

Héritage MovieClip > Classe UIObject > Classe UIComponent > Affichage > Composant Loader > Screen

Nom de classe ActionScript mx.screens.Screen

Les méthodes, propriétés et événements de la classe Screen vous permettent de créer et de manipuler des écrans lors de l'exécution.

Méthodes de la classe Screen

Le tableau suivant présente la méthode de la classe Screen.

Méthode	Description
`Screen.getChildScreen()`	Renvoie l'écran enfant de cet écran à un index spécifique.

Méthodes héritées de la classe UIObject

Le tableau suivant répertorie les méthodes de la classe Screen héritées de la classe UIObject. Pour appeler ces méthodes à partir de l'objet Screen, utilisez le formulaire *`ScreenInstance.methodName`*.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
`UIObject.setSize()`	Redimensionne l'objet à la taille demandée.
`UIObject.setSkin()`	Définit une enveloppe dans l'objet.
`UIObject.setStyle()`	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant répertorie les méthodes de la classe Screen héritées de la classe UIComponent. Pour appeler ces méthodes à partir de l'objet Screen, utilisez le formulaire *ScreenInstance.methodName*.

Méthode	Description
`UIComponent.getFocus()`	Renvoie une référence à l'objet ayant le focus.
`UIComponent.setFocus()`	Attribue le focus à l'occurrence de composant.

Méthodes héritées de la classe Loader

Le tableau suivant présente la méthode de la classe Screen héritée de la classe Loader. Pour appeler cette méthode à partir de l'objet Screen, utilisez le formulaire *ScreenInstance.methodName*.

Méthode	Description
`Loader.load()`	Charge le contenu spécifié par la propriété `contentPath`.

Propriétés de la classe Screen

Le tableau suivant répertorie les propriétés de la classe Screen.

Propriété	Description
`Screen.currentFocusedScreen`	Lecture seule : renvoie l'écran contenant le focus global actuel.
`Screen.indexInParent`	Lecture seule ; renvoie l'index de l'écran (dont la numérotation commence à zéro) dans la liste des écrans enfant de son écran parent.
`Screen.numChildScreens`	Lecture seule : renvoie le nombre d'écrans enfant contenus dans l'écran.
`Screen.parentIsScreen`	Lecture seule : renvoie une valeur booléenne (`true` ou `false`) indiquant si l'objet parent de l'écran est lui-même un écran.
`Screen.parentScreen`	Lecture seule : renvoie l'écran contenant l'écran spécifié.
`Screen.rootScreen`	Lecture seule : renvoie l'écran racine de l'arborescence (ou sous-arborescence) contenant l'écran.

Propriétés héritées de la classe UIObject

Le tableau suivant répertorie les propriétés de la classe Screen héritées de la classe UIObject. Pour accéder à ces propriétés à partir de l'objet Screen, utilisez le formulaire *`ScreenInstance.propertyName`*.

Propriété	Description
`UIObject.bottom`	Position du bord inférieur de l'objet par rapport au bord inférieur de son parent. Lecture seule.
`UIObject.height`	Hauteur de l'objet, en pixels. Lecture seule.
`UIObject.left`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.right`	Position du bord droit de l'objet par rapport au bord droit de son parent. Lecture seule.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Position du bord supérieur de l'objet par rapport à son parent. Lecture seule.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Largeur de l'objet, en pixels. Lecture seule.
`UIObject.x`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.y`	Bord supérieur de l'objet, en pixels. Lecture seule.

Propriétés héritées de la classe UIComponent

Le tableau suivant répertorie les propriétés de la classe Screen héritées de la classe UIComponent. Pour accéder à ces propriétés à partir de l'objet Screen, utilisez le formulaire *`ScreenInstance.propertyName`*.

Propriété	Description
`UIComponent.enabled`	Indique si le composant peut recevoir le focus et la saisie.
`UIComponent.tabIndex`	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Propriétés héritées de la classe Loader

Le tableau suivant répertorie les propriétés de la classe Screen héritées de la classe Loader. Pour accéder à ces propriétés à partir de l'objet Screen, utilisez le formulaire *ScreenInstance.propertyName*.

Propriété	Description
`Loader.autoLoad`	Valeur booléenne indiquant si le contenu se charge automatiquement (`true`) ou si vous devez appeler `Loader.load()` (`false`).
`Loader.bytesLoaded`	Propriété en lecture seule indiquant le nombre d'octets ayant été chargés.
`Loader.bytesTotal`	Propriété en lecture seule indiquant le nombres d'octets du contenu.
`Loader.content`	Référence au contenu du chargeur. Cette propriété est en lecture seule.
`Loader.contentPath`	Chaîne indiquant l'URL du contenu devant être chargé.
`Loader.percentLoaded`	Nombre indiquant le pourcentage de contenu chargé. Cette propriété est en lecture seule.
`Loader.scaleContent`	Valeur booléenne indiquant si le contenu est dimensionné pour s'ajuster au chargeur (`true`) ou si le chargeur est dimensionné pour s'ajuster au contenu (`false`).

Evénements de la classe Screen

Le tableau suivant répertorie les événements de la classe Screen.

Evénement	Description
`Screen.allTransitionsInDone`	Diffusé lorsque toutes les transitions « en entrée » appliquées à un écran sont terminées.
`Screen.allTransitionsOutDone`	Diffusé lorsque toutes les transitions « en sortie » appliquées à un écran sont terminées.
`Screen.mouseDown`	Diffusé lorsque l'utilisateur clique sur un objet (forme ou clip) appartenant directement à l'écran.
`Screen.mouseDownSomewhere`	Diffusé lorsque l'utilisateur clique quelque part sur la scène, mais pas nécessairement sur un objet appartenant à l'écran.
`Screen.mouseMove`	Diffusé lorsque le pointeur de la souris se déplace sur un écran.

Evénement	Description
Screen.mouseOut	Diffusé lorsque le pointeur de la souris se déplace de l'intérieur vers l'extérieur de l'écran.
Screen.mouseOver	Diffusé lorsque le pointeur de la souris se déplace de l'extérieur vers l'intérieur de l'écran.
Screen.mouseUp	Diffusé lorsque l'utilisateur relâche le bouton de la souris au-dessus d'un objet (forme ou clip) appartenant directement à l'écran.
Screen.mouseUpSomewhere	Diffusé lorsque l'utilisateur relâche le bouton de la souris quelque part sur la scène, mais pas nécessairement sur un objet appartenant à l'écran.

Evénements hérités de la classe UIObject

Le tableau suivant répertorie les événements de la classe Screen hérités de la classe UIObject.

Evénement	Description
UIObject.draw	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
UIObject.hide	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
UIObject.load	Diffusé lorsque des sous-objets sont créés.
UIObject.move	Diffusé lorsque l'objet a été déplacé.
UIObject.resize	Diffusé lorsqu'un objet a été redimensionné.
UIObject.reveal	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
UIObject.unload	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant répertorie les événements de la classe Screen hérités de la classe UIComponent.

Evénement	Description
UIComponent.focusIn	Diffusé lorsqu'un objet reçoit le focus.
UIComponent.focusOut	Diffusé lorsqu'un objet perd le focus.
UIComponent.keyDown	Diffusé lorsqu'une touche est enfoncée.
UIComponent.keyUp	Diffusé lorsqu'une touche est relâchée.

Evénements hérités de la classe Loader

Le tableau suivant répertorie les événements de la classe Screen hérités de la classe Loader.

Evénement	Description
`Loader.complete`	Déclenché à la fin du chargement du contenu.
`Loader.progress`	Déclenché pendant le chargement du contenu.

Screen.allTransitionsInDone

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
on(allTransitionsInDone) {
  // Votre code ici.
}
listenerObject = new Object();
listenerObject.allTransitionsInDone = function(eventObject){
  // Insertion du code ici.
}
screenObj.addEventListener("allTransitionsInDone", listenerObject)
```

Description

Evénement ; diffusé lorsque toutes les transitions « en entrée » appliquées à un écran sont terminées. L'événement `allTransitionsInDone` est diffusé par le gestionnaire de transition associé à *`screenObj`*.

Exemple

Dans l'exemple suivant, un bouton (`nextSlide_btn`) appartenant à la diapositive nommée `mySlide` apparaît lorsque toutes les transitions « in » appliquées à `mySlide` sont terminées.

```
// Associé à mySlide :
on(allTransitionsInDone) {
  this.nextSlide_btn._visible = true;
}
```

Voir aussi

`Screen.allTransitionsOutDone`

Screen.allTransitionsOutDone

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
on(allTransitionsOutDone) {
  // Votre code ici.
}
listenerObject = new Object();
listenerObject.allTransitionsOutDone = function(eventObject){
  // Insertion du code ici.
}
screenObj.addEventListener("allTransitionsOutDone", listenerObject)
```

Description

Evénement ; diffusé lorsque toutes les transitions « en sortie » appliquées à l'écran sont terminées. L'événement `allTransitionsOutDone` est diffusé par le gestionnaire de transition associé à *`screenObj`*.

Voir aussi

`Screen.currentFocusedScreen`

Screen.currentFocusedScreen

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`myScreen``.currentFocusedScreen`

Description

Propriété statique (lecture seule) ; renvoie une référence à l'objet d'écran le plus éloigné dans l'arborescence contenant le focus global actuel. *Leafmost* se réfère à l'écran le plus éloigné de l'écran racine dans la hiérarchie d'écrans. Le focus peut se trouver sur l'écran lui-même ou sur un clip, un objet texte ou un composant à l'intérieur de l'écran. Cette propriété prend la valeur `null` par défaut s'il n'y a aucun focus actuel.

Par exemple, supposons que vous avez une hiérarchie d'écrans d'exécution ayant l'aspect suivant :

```
presentation
  screen1
    subscreen1_1
      mymovieclip
        myUIButton

  screen2
    subscreen1_2
```

Si `myUIButton` a le focus, l'écran le plus éloigné contenant le focus est `subscreen1_1`, qui correspond à ce que `currentFocusedScreen` renvoie. Dans ce cas, `presentation`, `screen1` et `subscreen1_1` contiennent le focus, mais celui se trouvant « le plus près » (dans la hiérarchie d'écrans) des feuilles de l'arborescence (c'est-à-dire le plus loin de la racine) est `subscreen1_1`.

Exemple

L'exemple suivant affiche le nom de l'écran ayant actuellement le focus dans le panneau Sortie.

```
var currentFocus:mx.screens.Screen =
  mx.screens.Screen.currentFocusedScreen;
trace("Current screen is: " + currentFocus._name);
```

Screen.getChildScreen()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myScreen.getChildScreen(*childIndex*)

Paramètres

childIndex Nombre indiquant l'index (basé sur zéro) de l'écran enfant à renvoyer.

Renvoie

Un objet d'écran.

Description

Méthode : renvoie l'écran enfant de *myScreen* dont l'index correspond à *childIndex*.

Exemple

L'exemple suivant affiche dans le panneau Sortie les noms de tous les écrans enfant appartenant à l'écran racine nommé `Presentation`.

```
for (var i:Number = 0; i < _root.Presentation.numChildScreens; i++) {
    var childScreen:mx.screens.Screen =
  _root.Presentation.getChildScreen(i);
    trace(childScreen._name);
}
```

Screen.indexInParent

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myScreen.indexInParent

Description

Propriété (en lecture seule) ; contient l'index (basé sur zéro) de *myScreen* dans la liste des écrans enfant de son parent.

Exemple

L'exemple suivant affiche la position relative de l'écran myScreen dans la liste d'écrans enfant de son écran parent.

```
var numChildren:Number = myScreen._parent.numChildScreens;
var myIndex:Number = myScreen.indexInParent;
trace("I'm child slide # " + myIndex + " out of " + numChildren + "
  screens.");
```

Screen.mouseDown

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
on(mouseDown) {
  // Votre code ici.
}
listenerObject = new Object();
listenerObject.mouseDown = function(eventObj){
  // Insertion du code ici.
}
screenObj.addEventListener("mouseDown", listenerObject)
```

Description

Evénement : diffusé lorsque l'utilisateur clique sur un objet (par exemple, une forme ou un clip) appartenant directement à l'écran.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObj*) au gestionnaire. Chaque objet événement est doté de propriétés contenant des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

Le code suivant affiche dans le panneau Sortie le nom de l'écran ayant capturé l'événement souris.

```
on(mouseDown) {
  trace("Mouse down event on: " + eventObj.target._name);
}
```

Screen.mouseDownSomewhere

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
on(mouseDownSomewhere) {
  // Votre code ici.
}
listenerObject = new Object();
listenerObject.mouseDownSomewhere = function(eventObject){
  // Insertion du code ici.
}
screenObj.addEventListener("mouseDownSomewhere", listenerObject)
```

Description

Evénement ; diffusé lorsque l'utilisateur clique sur le bouton de la souris, mais pas nécessairement sur l'écran spécifié.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObj*) au gestionnaire. Chaque objet événement est doté de propriétés contenant des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Screen.mouseMove

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
on(mouseMove) {
  // Votre code ici.
}
listenerObject = new Object();
listenerObject.mouseMove = function(eventObject){
  // Insertion du code ici.
}
screenObj.addEventListener("mouseMove", listenerObject)
```

Description

Evénement : diffusé lorsque le pointeur de la souris se déplace sur l'écran. Cet événement est envoyé uniquement lorsque la souris se trouve sur le cadre de sélection de cet écran.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObj*) au gestionnaire. Chaque objet événement est doté de propriétés contenant des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

REMARQUE	Cet événement peut avoir une incidence sur les performances du système et doit donc être utilisé judicieusement.

Screen.mouseOut

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
on(mouseOut) {
  // Votre code ici.
}
listenerObject = new Object();
listenerObject.mouseOut = function(eventObject){
  // Insertion du code ici.
}
screenObj.addEventListener("mouseOut", listenerObject)
```

Description

Evénement ; diffusé lorsque le pointeur de la souris se déplace de l'intérieur vers l'extérieur du cadre de sélection de l'écran.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObj*) au gestionnaire. Chaque objet événement est doté de propriétés contenant des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

REMARQUE	Cet événement peut avoir une incidence sur les performances du système ; il doit donc être utilisé judicieusement.

Screen.mouseOver

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
on(mouseOver) {
  // Votre code ici.
}
listenerObject = new Object();
listenerObject.mouseOver = function(eventObject){
  // Insertion du code ici.
}
screenObj.addEventListener("mouseOver", listenerObject)
```

Description

Evénement ; diffusé lorsque le pointeur de la souris se déplace de l'extérieur vers l'intérieur du cadre de sélection de l'écran.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObj*) au gestionnaire. Chaque objet événement est doté de propriétés contenant des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

REMARQUE	Cet événement peut avoir une incidence sur les performances du système ; il doit donc être utilisé judicieusement.

Screen.mouseUp

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
on(mouseUp) {
  // Votre code ici.
}
listenerObject = new Object();
listenerObject.mouseUp = function(eventObject){
  // Insertion du code ici.
}
screenObj.addEventListener("mouseUp", listenerObject)
```

Description

Evénement : diffusé lorsque le bouton de la souris est relâché au-dessus de l'écran.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObj*) au gestionnaire. Chaque objet événement est doté de propriétés contenant des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Screen.mouseUpSomewhere

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
on(mouseUpSomewhere) {
  // Votre code ici.
}
listenerObject = new Object();
listenerObject.mouseUpSomewhere = function(eventObject){
  // Insertion du code ici.
}
screenObj.addEventListener("mouseUpSomewhere", listenerObject)
```

Description

Evénement ; diffusé lorsque l'utilisateur relâche le bouton de la souris, mais pas nécessairement sur l'écran spécifié.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObj*) au gestionnaire. Chaque objet événement est doté de propriétés contenant des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Screen.numChildScreens

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myScreen.numChildScreens

Description

Propriété (lecture seule) : renvoie le nombre d'écrans enfant contenus dans *myScreen*.

Exemple

L'exemple suivant affiche les noms de tous les écrans enfant appartenant à `myScreen`.

```
var howManyKids:Number = myScreen.numChildScreens;
for(i=0; i<howManyKids; i++) {
  var childScreen = myScreen.getChildScreen(i);
  trace(childScreen._name);
}
```

Voir aussi

Screen.getChildScreen()

Screen.parentIsScreen

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myScreen.parentIsScreen

Description

Propriété (lecture seule) : renvoie une valeur booléenne indiquant si l'objet parent de l'écran spécifié est également un écran (`true`) ou non (`false`). Si cette propriété est `false`, *myScreen* se trouve à la racine de sa hiérarchie d'écrans.

Exemple

Le code suivant détermine si l'objet parent de l'écran `myScreen` est également un écran. Si `myScreen.parentIsScreen` est `true`, une instruction `trace()` affiche le nombre d'écrans frères de `myScreen` dans le panneau Sortie. Si l'écran parent de `myScreen` n'est pas également un écran, Flash suppose que `myScreen` est la diapositive racine (maître) dans le diaporama et que par conséquent, elle n'a pas de frères.

```
if (myScreen.parentIsScreen) {
  trace("I have "+myScreen._parent.numChildScreens+" sibling screens");
} else {
  trace("I am the root screen and have no siblings");
}
```

Screen.parentScreen

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myScreen.parentScreen

Description

Propriété (lecture seule) : renvoie l'écran contenant *myScreen*. Renvoie `null` si *myScreen* est l'écran racine.

Exemple

L'exemple suivant affiche le nom de l'écran contenant l'écran `myScreen`.

```
var myParent:mx.screens.Screen = myScreen.rootScreen;
```

Screen.rootScreen

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myScreen.rootScreen

Description

Propriété (lecture seule) : renvoie l'écran situé en haut de la hiérarchie d'écrans et contenant *myScreen*.

Exemple

L'exemple suivant affiche le nom de l'écran racine contenant l'écran `myScreen`.

```
var myRoot:mx.screens.Screen = myScreen.rootScreen;
```

CHAPITRE 40

Composant ScrollPane

40

Le composant ScrollPane affiche des clips, des fichiers JPEG et SWF dans une zone que vous pouvez faire défiler. Le panneau défilant vous permet de limiter la portion d'écran occupée par ces types de médias. Il peut afficher du contenu chargé d'un disque local ou d'Internet. Vous pouvez définir ce contenu pendant la programmation et lors de l'exécution en utilisant ActionScript.

REMARQUE

Un composant ScrollPane est pris en charge pour ActionScript 2.0 et ActionScript 3.0. Ce document présente le composant de la version 2. Si vous utilisez celui de la version 3, reportez-vous à « Utilisation du composant ScrollPane » dans *Utilisation des composants ActionScript 3.0*.

Une fois que le panneau défilant a le focus, si son contenu présente des arrêts de tabulation valides, ces marqueurs reçoivent le focus. Après le dernier arrêt de tabulation dans le contenu, le focus passe au composant suivant. Les barres de défilement horizontale et verticale dans le panneau défilant ne reçoivent jamais le focus.

Une occurrence de ScrollPane reçoit le focus si un utilisateur clique dessus ou utilise les tabulations. Lorsqu'une occurrence de ScrollPane a le focus, vous pouvez utiliser les touches suivantes pour le contrôler :

Touche	Description
Flèche vers le bas	Le contenu se déplace d'une ligne de défilement verticale vers le haut.
Fin	Le contenu se déplace en bas du panneau défilant.
Flèche gauche	Le contenu se déplace d'une ligne de défilement horizontale vers la droite.
Page d'accueil	Le contenu se déplace en haut du panneau défilant.
Pg. Suiv.	Le contenu se déplace d'une page de défilement verticale vers le haut.
Pg. Préc.	Le contenu se déplace d'une page de défilement verticale vers le bas.
Flèche droite	Le contenu se déplace d'une ligne de défilement horizontale vers la gauche.
Flèche vers le haut	Le contenu se déplace d'une ligne de défilement verticale vers le bas.

Pour plus d'informations sur le contrôle du focus, reportez-vous à « Classe FocusManager », à la page 745 ou à « Création de la navigation personnalisée du focus » dans *Utilisation des composants ActionScript 2.0*.

L'aperçu en direct des occurrences de ScrollPane reflète les modifications apportées aux paramètres dans l'inspecteur des propriétés ou des composants pendant la programmation.

Utilisation du composant ScrollPane

Vous pouvez utiliser un panneau défilant pour afficher le contenu qui ne rentre pas dans la zone dans laquelle il est chargé. Par exemple, si vous devez afficher une image de grande taille mais que vous avez peu de place dans une application, vous pouvez la charger dans un panneau défilant.

Vous pouvez définir le paramètre `scrollDrag` d'un panneau défilant sur `true` afin de permettre aux utilisateurs de déplacer avec la souris le contenu à l'intérieur du panneau ; un curseur en forme de main apparaît sur le contenu. Contrairement à la plupart des autres composants, les événements sont diffusés lorsque le bouton de la souris est enfoncé et continuent à diffuser jusqu'à ce que l'utilisateur relâche le bouton de la souris. Si le contenu d'un panneau défilant comporte des arrêts de tabulation valides, vous devez définir `scrollDrag` sur `false` sinon chaque interaction de la souris avec les contenus invoquera le déplacement par défilement.

Les composants tels que Loader, ScrollPane et Window ont des événements pour déterminer à quel moment le chargement du contenu est terminé. Ainsi, si vous souhaitez définir des propriétés sur le contenu d'un composant Loader, ScrollPane ou Window, ajoutez l'instruction de propriété dans un gestionnaire d'événements « complete », comme indiqué dans l'exemple suivant : Considérons l'exemple suivant :

```
loadtest = new Object();
loadtest.complete = function(eventObject){
  content_mc._width= 100;
}
my_scrollpane.addEventListener("complete", loadtest)
```

Pour plus d'informations, voir la section « ScrollPane.content », à la page 1153.

Paramètres de ScrollPane

Dans l'inspecteur des propriétés ou l'inspecteur de composants (option de menu Fenêtre > Inspecteur de composants), vous pouvez définir les paramètres de création suivants pour chaque occurrence ScrollPane :

contentPath indique le contenu à charger dans le panneau défilant. Cette valeur peut être un chemin relatif vers un fichier local SWF ou JPEG, ou un chemin relatif ou absolu vers un fichier sur Internet. Il peut aussi s'agir de l'identificateur de liaison du symbole d'un clip dans la bibliothèque, défini sur Exporter pour ActionScript.

hLineScrollSize indique de combien d'unités se déplace une barre de défilement horizontale chaque fois que vous cliquez sur un bouton fléché. La valeur par défaut est 5.

hPageScrollSize indique de combien d'unités se déplace une barre de défilement horizontale chaque fois que vous cliquez sur le rail. La valeur par défaut est 20.

hScrollPolicy affiche les barres de défilement horizontales. Les valeurs possibles sont : `on`, `off` ou `auto`. La valeur par défaut est `auto`.

scrollDrag est une valeur booléenne qui détermine si le défilement a lieu (`true`) ou non (`false`) lorsqu'un utilisateur fait glisser le contenu dans le panneau défilant. La valeur par défaut est `false`.

vLineScrollSize indique de combien d'unités se déplace une barre de défilement verticale chaque fois que vous cliquez sur un bouton fléché. La valeur par défaut est 5.

vPageScrollSize indique de combien d'unités se déplace une barre de défilement verticale chaque fois que vous cliquez sur le rail d'une barre de défilement. La valeur par défaut est 20.

vScrollPolicy affiche les barres de défilement verticales. Les valeurs possibles sont : `on`, `off` ou `auto`. La valeur par défaut est `auto`.

Dans l'inspecteur des composants, vous pouvez définir les paramètres supplémentaires suivants pour chaque occurrence du composant ScrollPane (Fenêtre > Inspecteur de composants) :

enabled est une valeur booléenne qui indique si le composant peut recevoir le focus et l'entrée. La valeur par défaut est `true`.

visible est une valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`). La valeur par défaut est `true`.

REMARQUE	Les propriétés minHeight et minWidth sont utilisées par des routines de dimensionnement internes. Elles sont définies dans UIObject et remplacées par différents composants selon vos besoins. Ces propriétés peuvent être utilisées pour créer un gestionnaire de dispositions personnalisées. Dans le cas contraire, la définition de ces propriétés dans l'inspecteur de composants n'a aucun effet visible.

Vous pouvez rédiger du code ActionScript pour contrôler ces options et d'autres options d'un composant ScrollPane en utilisant ses propriétés, méthodes et événements. Pour plus d'informations, voir « Classe ScrollPane », à la page 1146.

Création d'une application avec le composant ScrollPane

La procédure suivante explique comment ajouter un composant ScrollPane à une application pendant la programmation. Dans cet exemple, le panneau défilant charge une image depuis un chemin spécifié dans la propriété `contentPath`.

Pour créer une application avec le composant ScrollPane :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser le composant ScrollPane du panneau Composants jusqu'à la bibliothèque.
3. Sélectionnez l'image 1 dans le scénario principal, ouvrez le panneau Actions et saisissez le code suivant :

```
/**
 Requiert :
  - ScrollPane dans la bibliothèque
*/

System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(320, 240);

// Création d'un objet écouteur pour la position de défilement vertical.
var scrollListener:Object = new Object();
scrollListener.scroll = function(evt_obj:Object) {
 trace("hPosition: " + my_sp.hPosition + ", vPosition = " +
  my_sp.vPosition);
};
// Ajout de l'écouteur.
my_sp.addEventListener("scroll", scrollListener);

// Création de l'objet écouteur pour le chargement terminé.
var completeListener:Object = new Object();
completeListener.complete = function(evt_obj:Object) {
 trace(evt_obj.target.contentPath + " has completed loading.");
};
// Ajout de l'écouteur.
my_sp.addEventListener("complete", completeListener);

my_sp.contentPath = "http://www.helpexamples.com/flash/images/
  image1.jpg";
```

L'exemple crée un panneau défilant, définit sa taille et y charge une image à l'aide de la propriété `contentPath`. Il crée également deux écouteurs. Le premier écoute un événement scroll et affiche la position de l'image lorsque l'utilisateur utilise le défilement vertical ou horizontal. Le deuxième écoute un événement complete et affiche un message dans le panneau Sortie qui indique que le chargement de l'image est terminé.

Personnalisation du composant ScrollPane

Vous pouvez transformer un composant ScrollPane horizontalement et verticalement pendant la programmation et lors de l'exécution. Lors de la programmation, sélectionnez le composant sur la scène et utilisez l'outil Transformer librement ou une commande Modification > Transformer. Lors de l'exécution, utilisez la méthode `setSize()` (reportez-vous à `UIObject.setSize()`) ou toute propriété et méthode applicable de la classe ScrollPane.

N'oubliez pas les points suivants concernant le composant ScrollPane :

- Le composant ScrollPane définit le coin supérieur gauche comme point d'alignement de son contenu.
- Lorsque la barre de défilement horizontale est désactivée, la barre de défilement verticale s'affiche de haut en bas sur le côté droit du panneau défilant. Lorsque la barre de défilement verticale est désactivée, la barre de défilement horizontale s'affiche de gauche à droite en bas du panneau défilant. Vous pouvez également désactiver ces deux barres.
- Si le panneau défilant n'est pas assez grand, il se peut que le contenu ne s'affiche pas correctement.
- Lorsque le panneau défilant est redimensionné, les boutons conservent la même taille. Le rail et le curseur de défilement sont agrandis ou réduits et leurs zones réactives sont redimensionnées.

Utilisation de styles avec le composant ScrollPane

Le composant ScrollPane prend en charge les styles suivants :

Style	Thème	Description
`themeColor`	Halo	Couleur de base d'un composant. `"haloGreen"`, `"haloBlue"` et `"haloOrange"` sont des valeurs possibles. La valeur par défaut est `"haloGreen"`.
`borderStyle`	Les deux	Le composant ScrollPane utilise une occurrence de RectBorder en tant que bordure et répond aux styles définis sur cette classe. Voir « Classe RectBorder », à la page 1111. Le style de bordure par défaut est `"inset"`.

Style	Thème	Description
scrollTrackColor	Sample	Couleur d'arrière-plan du rail de défilement. La valeur par défaut est 0xCCCCCC (gris clair).
symbolColor	Sample	La couleur des flèches sur les boutons de la barre de défilement. La valeur par défaut est 0x000000 (noir).
symbolDisabledColor	Sample	La couleur des flèches désactivées sur les boutons de la barre de défilement. La valeur par défaut est 0x848384 (gris foncé).

Utilisation d'enveloppes avec le composant ScrollPane

Le composant ScrollPane utilise une occurrence de RectBorder pour sa bordure et des barres de défilement pour des éléments de défilement. Pour plus d'informations sur l'application d'enveloppe à ces éléments visuels, reportez-vous à « Classe RectBorder », à la page 1111 et à « Utilisation d'enveloppes avec le composant UIScrollBar », à la page 1448.

Classe ScrollPane

Héritage MovieClip > Classe UIObject > Classe UIComponent > View > ScrollView > ScrollPane

Nom de classe ActionScript mx.containers.ScrollPane

Les propriétés de la classe ScrollPane permettent d'effectuer les opérations suivantes au moment de l'exécution : définir le contenu, contrôler la progression du chargement et régler la quantité d'informations à faire défiler.

La définition d'une propriété de la classe ScrollPane avec ActionScript annule le paramètre du même nom défini dans l'inspecteur des propriétés ou des composants.

Vous pouvez configurer un panneau défilant de façon à ce que les utilisateurs puissent y faire défiler le contenu. Pour ce faire, définissez la propriété `scrollDrag` sur `true` ; un curseur en forme de main apparaît sur le contenu. Contrairement à la plupart des autres composants, les événements sont diffusés lorsque le bouton de la souris est enfoncé et continuent à diffuser jusqu'à ce que l'utilisateur relâche le bouton de la souris. Si le contenu d'un panneau défilant comporte des arrêts de tabulation valides, vous devez définir `scrollDrag` sur `false` sinon chaque interaction de la souris avec les contenus invoquera le déplacement par défilement.

Toutes les classes de composants ont une propriété `version` qui correspond à une propriété de classe. Les propriétés de classe ne sont disponibles que pour la classe elle-même. La propriété `version` renvoie une chaîne qui indique la version du composant. Pour accéder à cette propriété, utilisez le code suivant :

```
trace(mx.containers.ScrollPane.version);
```

REMARQUE

Le code `trace(myScrollPaneInstance.version);` renvoie `undefined`.

Méthodes de la classe ScrollPane

Le tableau suivant répertorie les méthodes de la classe ScrollPane.

Méthode	Description
`ScrollPane.getBytesLoaded()`	Renvoie le nombre d'octets du contenu chargé.
`ScrollPane.getBytesTotal()`	Renvoie le nombre total d'octets du contenu à charger.
`ScrollPane.refreshPane()`	Recharge le contenu du panneau défilant (mais ne redessine pas la barre de défilement).

Méthodes héritées de la classe UIObject

Le tableau suivant répertorie les méthodes de la classe ScrollPane héritées de la classe UIObject. Lors de l'appel à ces méthodes depuis l'objet ScrollPane, utilisez le formulaire *`ScrollPaneInstance.methodName`*.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet à l'emplacement demandé.

Méthode	Description
UIObject.redraw()	Force la validation de l'objet pour qu'il soit dessiné dans l'image active.
UIObject.setSize()	Redimensionne l'objet à la taille demandée.
UIObject.setSkin()	Définit une enveloppe dans l'objet.
UIObject.setStyle()	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant répertorie les méthodes de la classe ScrollPane héritées de la classe UIComponent. Lors de l'appel à ces méthodes depuis l'objet ScrollPane, utilisez le formulaire *ScrollPaneInstance.methodName*.

Méthode	Description
UIComponent.getFocus()	Renvoie une référence à l'objet ayant le focus.
UIComponent.setFocus()	Attribue le focus à l'occurrence de composant.

Propriétés de la classe ScrollPane

Le tableau suivant répertorie les propriétés de la classe ScrollPane.

Méthode	Description
ScrollPane.content	Référence au contenu chargé dans le panneau défilant (lecture seule).
ScrollPane.contentPath	Chaîne indiquant une URL absolue ou relative du fichier SWF ou JPEG à charger dans le panneau défilant, ou qui correspond à l'identifiant de liaison d'un clip dans la bibliothèque du document en cours.
ScrollPane.hLineScrollSize	Quantité de contenu à faire défiler horizontalement lorsque vous cliquez sur un bouton fléché.
ScrollPane.hPageScrollSize	Quantité de contenu à faire défiler horizontalement lorsque vous cliquez sur le rail de défilement.
ScrollPane.hPosition	Position horizontale (en pixels) de la barre de défilement horizontale du panneau défilant.

Méthode	Description
`ScrollPane.hScrollPolicy`	Etat de la barre de défilement horizontale. Elle peut être toujours activée (`"on"`), toujours désactivée (`"off"`) ou disponible en fonction des besoins (`"auto"`). La valeur par défaut est `"auto"`.
`ScrollPane.scrollDrag`	Indique si le défilement a lieu (`true`) ou non (`false`) lorsqu'un utilisateur fait glisser du contenu dans le panneau défilant. La valeur par défaut est `false`.
`ScrollPane.vLineScrollSize`	Quantité de contenu à faire défiler verticalement lorsque vous cliquez sur un bouton fléché.
`ScrollPane.vPageScrollSize`	Quantité de contenu à faire défiler verticalement lorsque vous cliquez sur le rail de défilement.
`ScrollPane.vPosition`	Position horizontale (en pixels) de la barre de défilement verticale du panneau défilant.
`ScrollPane.vScrollPolicy`	Etat de la barre de défilement verticale. Elle peut être toujours activée (`"on"`), toujours désactivée (`"off"`) ou disponible en fonction des besoins (`"auto"`). La valeur par défaut est `"auto"`.

Propriétés héritées de la classe UIObject

Le tableau suivant répertorie les propriétés de la classe ScrollPane héritées de la classe UIObject. Lors de l'appel à ces propriétés depuis l'objet ScrollPane, utilisez le formulaire *`ScrollPaneInstance.propertyName`*.

Propriété	Description
`UIObject.bottom`	Lecture seule ; position du bord inférieur de l'objet par rapport au bord inférieur de son parent.
`UIObject.height`	Lecture seule ; hauteur de l'objet, en pixels.
`UIObject.left`	Lecture seule ; bord gauche de l'objet, en pixels.
`UIObject.right`	Lecture seule ; position du bord droit de l'objet par rapport au bord droit de son parent.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Lecture seule ; renvoie la position du bord supérieur de l'objet par rapport à son parent.

Propriété	Description
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Lecture seule ; largeur de l'objet, en pixels.
`UIObject.x`	Lecture seule ; bord gauche de l'objet, en pixels.
`UIObject.y`	Lecture seule ; bord supérieur de l'objet, en pixels.

Propriétés héritées de la classe UIComponent

Le tableau suivant répertorie les propriétés de la classe ScrollPane héritées de la classe UIComponent. Lors de l'appel à ces propriétés depuis l'objet ScrollPane, utilisez le formulaire *`ScrollPaneInstance.propertyName`*.

Propriété	Description
`UIComponent.enabled`	Indique si le composant peut recevoir le focus et la saisie.
`UIComponent.tabIndex`	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Evénements de la classe ScrollPane

Le tableau suivant répertorie les événements de la classe ScrollPane.

Evénement	Description
`ScrollPane.complete`	Diffusé lorsque le contenu du panneau défilant est chargé.
`ScrollPane.progress`	Diffusé lorsque le contenu du panneau défilant est en cours de chargement.
`ScrollPane.scroll`	Diffusé lorsque vous cliquez sur la barre de défilement.

Evénements hérités de la classe UIObject

Le tableau suivant répertorie les événements de la classe ScrollPane hérités de la classe UIObject.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.

Evénement	Description
UIObject.move	Diffusé lorsque l'objet a été déplacé.
UIObject.resize	Diffusé lorsqu'un objet a été redimensionné.
UIObject.reveal	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
UIObject.unload	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant répertorie les événements de la classe ScrollPane hérités de la classe UIComponent.

Evénement	Description
UIComponent.focusIn	Diffusé lorsqu'un objet reçoit le focus.
UIComponent.focusOut	Diffusé lorsqu'un objet perd le focus.
UIComponent.keyDown	Diffusé lorsqu'une touche est enfoncée.
UIComponent.keyUp	Diffusé lorsqu'une touche est relâchée.

ScrollPane.complete

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.complete = function(eventObject:Object) {
  // ...
};
scrollPaneInstance.addEventListener("complete", listenerObject);
```

Utilisation 2 :

```
on (complete) {
  //...
}
```

Description

Evénement : diffusé à tous les écouteurs enregistrés une fois le contenu chargé.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence de composant (*scrollPaneInstance*) distribue un événement (ici, complete) qui est géré par une fonction (également appelée *gestionnaire*) dans un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée au déclenchement de l'événement. Lorsqu'il est déclenché, l'événement transmet automatiquement un objet événement (*eventObject*) à la méthode de l'objet écouteur. Chaque objet événement est doté de propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode EventDispatcher.addEventListener() sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel à un gestionnaire on() et doit être associé directement à une occurrence ScrollPane. Le mot-clé this, utilisé dans un gestionnaire on() lié à un composant, correspond à l'occurrence du composant. Par exemple, le code suivant, associé à l'occurrence ScrollPane myScrollPaneComponent, envoie « _level0.myScrollPaneComponent » dans le panneau Sortie :

```
on (complete) {
  trace(this);
}
```

Exemple

L'exemple suivant crée un objet écouteur avec un gestionnaire d'événement complete pour l'occurrence de ScrollPane. Lorsque le contenu du panneau défilant est chargé, l'écouteur affiche un message dans le panneau Sortie.

Vous faites d'abord glisser le composant ScrollPane du panneau Composants à la bibliothèque puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - ScrollPane dans la bibliothèque
*/

System.security.allowDomain("http://www.helpexamples.com");

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(320, 240);
```

```
// Création de l'objet écouteur pour le chargement terminé.
var completeListener:Object = new Object();
completeListener.complete = function(evt_obj:Object) {
 trace(evt_obj.target.contentPath + " has completed loading.");
};
// Ajout de l'écouteur.
my_sp.addEventListener("complete", completeListener);

my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
```

ScrollPane.content

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

scrollPaneInstance.content

Description

Propriété en lecture seule ; référence au contenu du panneau défilant. La valeur est `undefined` jusqu'à ce que le chargement commence.

Exemple

Cet exemple définit la propriété `contentPath` pour charger un panneau défilant avec une image (ou techniquement, un clip contenant une image JPEG). Il crée également un incrémenteur numérique que l'utilisateur peut incrémenter ou décrémenter de 10, jusqu'à une valeur de 100. Lorsque l'utilisateur change la valeur dans le NumericStepper, un écouteur définit la transparence (`content._alpha`) de l'image sur le pourcentage spécifié. Notez que `_alpha` est une propriété MovieClip.

Vous faites d'abord glisser les composants ScrollPane et NumericStepper du panneau Composants à la bibliothèque du document actuel puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - ScrollPane dans la bibliothèque
  - NumericStepper dans la bibliothèque
*/

System.security.allowDomain("http://www.helpexamples.com");
```

```
this.createClassObject(mx.controls.NumericStepper, "my_nstep", 10,
  {minimum:10, maximum:100, stepSize:10});
my_nstep.value = my_nstep.maximum;

this.createClassObject(mx.containers.ScrollPane, "my_sp", 20);
my_sp.move(0, 30);
my_sp.setSize(180, 160);
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image2.jpg";

var nstepListener:Object = new Object();
nstepListener.change = function(evt_obj:Object) {
 my_sp.content._alpha = my_nstep.value;
}
my_nstep.addEventListener("change", nstepListener);
```

Voir aussi

`ScrollPane.contentPath`

ScrollPane.contentPath

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`scrollPaneInstance``.contentPath`

Description

Propriété : chaîne qui indique une URL absolue ou relative du fichier SWF ou JPEG à charger dans le panneau défilant. Un chemin relatif doit être relatif au fichier SWF chargeant le contenu.

Si vous chargez le contenu en utilisant une URL relative, le contenu chargé doit être relatif à l'emplacement du fichier SWF contenant le panneau défilant. Par exemple, une application utilisant un composant ScrollPane qui réside dans le répertoire /scrollpane/nav/example.swf pourrait charger les contenus à partir du répertoire /scrollpane/content/flash/logo.swf à l'aide de la propriété `contentPath` suivante : `"../content/flash/logo.swf"`

Exemple

L'exemple suivant indique comment définir la propriété `contentPath` pour charger un composant ScrollPane depuis trois sources différentes : 1) une image sur Internet ; 2) un clip dans la bibliothèque ; 3) un fichier SWF du répertoire de travail actuel. Utilisez une seule source à la fois.

Vous faites d'abord glisser le composant ScrollPane du panneau Composants à la bibliothèque du document actuel. Pour essayer l'option 2, vous devez créer un clip dans la bibliothèque et faire référence à son nom. Pour essayer l'option 3, créez un fichier SWF dans le répertoire de travail actuel et spécifiez son nom. Ajoutez ensuite le code suivant à l'image 1 :

```
/**
 Requiert :
  - ScrollPane sur la scène (nom d'occurrence : my_sp)
  - Symbole avec identificateur de liaison de « movieClip_Name » dans la
  bibliothèque ** facultatif
  - Fichier logo.swf dans le répertoire de travail ** facultatif
*/

System.security.allowDomain("http://www.helpexamples.com");

var my_sp:mx.containers.ScrollPane;

// Méthode 1 : image JPEG
my_sp.contentPath ="http://www.helpexamples.com/flash/images/image1.jpg";

// Méthode 2 : Symbole dans la bibliothèque
my_sp.contentPath ="movieClip_Name";

// Méthode 3 : Fichier SWF
my_sp.contentPath ="logo.swf";
```

Voir aussi

`ScrollPane.content`

ScrollPane.getBytesLoaded()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`scrollPaneInstance``.getBytesLoaded()`

Paramètres

Aucun.

Valeur renvoyée

Le nombre d'octets chargés dans le panneau défilant.

Description

Méthode : renvoie le nombre d'octets chargés dans l'occurrence de ScrollPane. Vous pouvez appeler cette méthode à intervalles réguliers pendant le chargement du contenu afin d'en vérifier la progression.

Exemple

Cet exemple crée une occurrence ScrollPane appelée `my_sp` et définit un objet écouteur appelé `loadListener` avec un gestionnaire d'événements `progress`. Le gestionnaire d'événements appelle les fonctions `getBytesLoaded()` et `getBytesTotal()` pour afficher la progression du chargement dans le panneau Sortie.

Faites d'abord glisser le composant ScrollPane du panneau Composants vers la bibliothèque du document en cours, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant ScrollPane dans la bibliothèque
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

var loadListener:Object = new Object();
loadListener.progress = function(evt_obj:Object) {
 trace(my_sp.getBytesLoaded() + " of " + my_sp.getBytesTotal() + " bytes
  loaded.");
};
my_sp.addEventListener("progress", loadListener);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
```

ScrollPane.getBytesTotal()

Disponibilité
Flash Player 6 (6.0.79.0).

Edition
Flash MX 2004.

Utilisation
scrollPaneInstance.getBytesTotal()

Paramètres
Aucun.

Valeur renvoyée
Un nombre.

Description
Méthode : renvoie le nombre total d'octets à charger dans l'occurrence de ScrollPane.

Exemple
Cet exemple crée une occurrence ScrollPane appelée `my_sp` et définit un objet écouteur appelé `loadListener` avec un gestionnaire d'événements `progress`. Le gestionnaire d'événements appelle les fonctions `getBytesLoaded()` et `getBytesTotal()` pour afficher la progression du chargement dans le panneau Sortie.

Faites d'abord glisser le composant ScrollPane du panneau Composants vers la bibliothèque du document en cours, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant ScrollPane dans la bibliothèque
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

var loadListener:Object = new Object();
loadListener.progress = function(evt_obj:Object) {
 trace(my_sp.getBytesLoaded() + " of " + my_sp.getBytesTotal() + " bytes
  loaded.");
};
my_sp.addEventListener("progress", loadListener);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
```

Voir aussi
ScrollPane.getBytesLoaded()

ScrollPane.hLineScrollSize

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

scrollPaneInstance.hLineScrollSize

Description

Propriété : nombre indiquant de combien de pixels se déplace le contenu lorsque vous cliquez sur une flèche de la barre de défilement horizontale. La valeur par défaut est 5.

Exemple

Cet exemple crée une occurrence de composant ScrollPane appelée `my_sp`, la charge avec une image et définit la propriété `hLineScrollSize` pour faire défiler 100 pixels lorsque l'utilisateur clique sur une flèche sur la barre de défilement horizontal.

Faites d'abord glisser le composant ScrollPane du panneau Composants vers la bibliothèque du document en cours, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant ScrollPane dans la bibliothèque
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
// Défilement de 100 pixels lorsque l'utilisateur clique sur des flèches
// de la barre de défilement horizontal.
my_sp.hLineScrollSize = 100;
```

ScrollPane.hPageScrollSize

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

scrollPaneInstance.hPageScrollSize

Description

Propriété : nombre indiquant de combien de pixels se déplace le contenu lorsque vous cliquez sur le rail de la barre de défilement horizontale. La valeur par défaut est 20.

Exemple

Cet exemple crée une occurrence de composant ScrollPane appelée `my_sp`, la charge avec une image et définit la propriété `hPageScrollSize` pour faire défiler 100 pixels lorsque l'utilisateur clique à l'intérieur de la barre de défilement horizontal.

Vous faites d'abord glisser le composant ScrollPane du panneau Composants à la bibliothèque du document actuel puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant ScrollPane dans la bibliothèque
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";

// Défilement de 100 pixels lorsque l'utilisateur clique sur la barre
// de défilement horizontal.
my_sp.hPageScrollSize = 100;
```

ScrollPane.hPosition

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

scrollPaneInstance.hPosition

Description

Propriété ; oriente le contenu du panneau de défilement en pixels et ajuste le curseur de défilement horizontal proportionnellement. La position 0 se trouve à l'extrême gauche du rail de défilement, ce qui permet de rendre visible le bord gauche du contenu du panneau défilant dans ce dernier.

Exemple

Cet exemple crée une occurrence ScrollPane appelée `my_sp`, la charge avec une image et crée un écouteur pour gérer l'événement scroll et afficher les positions de défilement horizontal (`hPosition`) et vertical (`vPosition`) lorsque l'utilisateur clique sur la barre de défilement.

Faites d'abord glisser le composant ScrollPane du panneau Composants jusqu'à la bibliothèque du document actif, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant ScrollPane dans la bibliothèque
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
// Défilement de 100 pixels lorsque l'utilisateur clique sur la barre
// de défilement horizontal.
my_sp.hPageScrollSize = 100;

// Création d'un objet écouteur.
var spListener:Object = new Object();
spListener.scroll = function(evt_obj:Object) {
   trace("hPosition = " + my_sp.hPosition + ", vPosition = " +
  my_sp.vPosition);
}
// Ajout de l'écouteur.
my_sp.addEventListener("scroll", spListener);
```

ScrollPane.hScrollPolicy

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

scrollPaneInstance.hScrollPolicy

Description

Propriété : détermine si la barre de défilement horizontale est toujours présente (`"on"`), jamais présente (`"off"`) ou s'affiche automatiquement en fonction de la taille de l'image (`"auto"`). La valeur par défaut est `"auto"`.

Exemple

L'exemple suivant crée une occurrence d'un composant ScrollPane appelée `my_sp`, définit `hScrollPolicy` sur `off` pour que la barre de défilement horizontal n'apparaisse pas, et la charge avec une image.

Faites d'abord glisser le composant ScrollPane du panneau Composants vers la bibliothèque du document en cours, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant ScrollPane dans la bibliothèque
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);
my_sp.hScrollPolicy = "off";

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
```

ScrollPane.progress

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.progress = function(eventObject:Object) {
  // ...
};
scrollPaneInstance.addEventListener("progress", listenerObject);
```

Utilisation 2 :

```
on (progress) {
  // ...
}
```

Description

Evénement : diffusé à l'ensemble des écouteurs enregistrés pendant le chargement du contenu. L'événement progress n'est pas toujours diffusé. L'événement `complete` peut être diffusé sans qu'aucun événement `progress` ne soit distribué. Ceci peut particulièrement se produire si le contenu chargé est un fichier local. Votre application déclenche l'événement `progress` au début du chargement du contenu en définissant la valeur de la propriété `contentPath`.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence de composant (*`scrollPaneInstance`*) distribue un événement (ici, `progress`) qui est géré par une fonction (également appelée *gestionnaire*) dans un objet écouteur (*`listenerObject`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée au déclenchement de l'événement. Lorsqu'il est déclenché, l'événement transmet automatiquement un objet événement (*`eventObject`*) à la méthode de l'objet écouteur. Chaque objet événement est doté de propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence ScrollPane. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence du composant. Par exemple, le code suivant, associé à l'occurrence du composant ScrollPane `mySPComponent`, envoie « _level0.mySPComponent » dans le panneau Sortie :

```
on (progress) {
  trace(this);
}
```

Exemple

Cet exemple crée une occurrence de composant ScrollPane appelée `my_sp` et définit un objet écouteur appelé `spListener` avec un gestionnaire d'événements `progress`. Le gestionnaire d'événements appelle les fonctions `getBytesLoaded()` et `getBytesTotal()` pour afficher la progression du chargement dans le panneau Sortie.

Faites d'abord glisser le composant ScrollPane du panneau Composants vers la bibliothèque du document en cours, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant ScrollPane dans la bibliothèque
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

var spListener:Object = new Object();
spListener.progress = function(evt_obj:Object):Void {
 trace("Loading " + my_sp.contentPath);
 trace(my_sp.getBytesLoaded() + " of " + my_sp.getBytesTotal() + " bytes
  loaded");
};
my_sp.addEventListener("progress", spListener);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
```

ScrollPane.refreshPane()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

scrollPaneInstance.refreshPane()

Paramètres

Aucun.

Valeur renvoyée

Aucune.

Description

Méthode : actualise le panneau défilant après le chargement du contenu. Cette méthode recharge le contenu, mais ne réinitialise pas la barre de défilement. Vous pouvez utiliser cette méthode si, par exemple, vous avez chargé un formulaire dans un panneau défilant et qu'une propriété d'entrée (par exemple, dans un champ de texte) a été modifiée par ActionScript. Dans ce cas, appelez `refreshPane()` pour recharger le même formulaire avec les nouvelles valeurs de propriétés d'entrée.

Exemple

Cet exemple crée un bouton Refresh (Actualiser) et une occurrence de composant ScrollPane appelée `my_sp`. Il charge le composant ScrollPane avec une image et crée un écouteur pour un événement click sur le bouton. Lorsqu'un événement click se produit, l'exemple appelle la fonction `refreshPane()`, qui recharge le contenu du panneau défilant.

Faites d'abord glisser le composant ScrollPane du panneau Composants jusqu'à la bibliothèque du document actif, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant ScrollPane dans la bibliothèque
*/

this.createClassObject(mx.controls.Button, "my_button", 10,
  {label:"Refresh"});
this.createClassObject(mx.containers.ScrollPane, "my_sp", 20);
my_sp.move(0, 30);
my_sp.setSize(360, 280);

var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object) {
 my_sp.refreshPane();
}
my_button.addEventListener("click", buttonListener);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
```

ScrollPane.scroll

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.scroll = function(eventObject:Object):Void {
  // ...
};
scrollPaneInstance.addEventListener("scroll", listenerObject);
```

Utilisation 2 :

```
on (scroll) {
  // ...
}
```

Objet événement

Outre les propriétés d'objet événement standard, deux propriétés supplémentaires sont définies pour l'événement `scroll` : une propriété `type` dont la valeur est `"scroll"` et une propriété `direction` dont la valeur peut être `"vertical"` ou `"horizontal"`.

Outre les propriétés d'objet événement standard, deux propriétés supplémentaires sont définies pour l'événement `ProgressBar.progress` : `current` (la valeur chargée qui est égale au `total`) et `total` (la valeur totale).

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsqu'un utilisateur clique sur les boutons, le curseur de défilement ou le rail de la barre de défilement. Contrairement aux autres événements, l'événement `scroll` est diffusé lorsqu'un utilisateur clique sur le bouton de la souris sur la barre de défilement et continue la diffusion jusqu'à ce qu'un utilisateur relâche la souris.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence de composant (*`scrollPaneInstance`*) distribue un événement (ici, `scroll`) qui est géré par une fonction (également appelée *gestionnaire*) dans un objet écouteur (*`listenerObject`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée au déclenchement de l'événement. Lorsqu'il est déclenché, l'événement transmet automatiquement un objet événement (*`eventObject`*) à la méthode de l'objet écouteur. Chaque objet événement est doté de propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence ScrollPane. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence du composant. Par exemple, le code suivant, associé à l'occurrence `my_sp`, envoie « _level0.my_sp » vers le panneau Sortie :

```
on (scroll) {
  trace(this);
}
```

Exemple

Cet exemple crée une occurrence de composant ScrollPane appelée `my_sp`, la charge avec une image, et crée un écouteur pour l'événement scroll. Lorsqu'un événement scroll a lieu, l'exemple affiche les positions de défilement horizontal (`hPosition`) et vertical (`vPosition`) dans le panneau Sortie.

Faites d'abord glisser le composant ScrollPane du panneau Composants jusqu'à la bibliothèque du document actif, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant ScrollPane dans la bibliothèque
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
// Défilement de 100 pixels lorsque l'utilisateur clique sur la barre
// horizontale.
my_sp.hPageScrollSize = 100;

// Création d'un objet écouteur.
var spListener:Object = new Object();
spListener.scroll = function(evt_obj:Object):Void {
   trace("hPosition = " + my_sp.hPosition + ", vPosition = " +
  my_sp.vPosition);
};
// Ajout de l'écouteur.
my_sp.addEventListener("scroll", spListener);
```

Voir aussi

`EventDispatcher.addEventListener()`

ScrollPane.scrollDrag

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

scrollPaneInstance.scrollDrag

Description

Propriété : valeur booléenne qui indique si le défilement a lieu (`true`) ou non (`false`) lorsqu'un utilisateur fait glisser du contenu dans le panneau défilant. La valeur par défaut est `false`.

Exemple

Cet exemple crée une occurrence de composant ScrollPane appelée `my_sp`, la charge avec une image et définit la propriété `scrollDrag` sur `true`, permettant ainsi à l'utilisateur d'utiliser le défilement en faisant glisser l'image dans le panneau défilant.

Faites d'abord glisser le composant ScrollPane du panneau Composants jusqu'à la bibliothèque du document actif, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant ScrollPane dans la bibliothèque
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";

// Activation du défilement en faisant glisser le panneau défilant.
my_sp.scrollDrag = true;
```

ScrollPane.vLineScrollSize

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

scrollPaneInstance.vLineScrollSize

Description

Property : nombre indiquant de combien de pixels le contenu se déplace dans la zone d'affichage lorsque l'utilisateur clique sur une flèche de défilement dans une barre de défilement verticale. La valeur par défaut est 5.

Exemple

Cet exemple crée une occurrence ScrollPane appelée `my_sp`, la charge avec une image et définit la propriété `vLineScrollSize` pour faire défiler 20 pixels lorsque l'utilisateur clique sur une flèche dans la barre de défilement vertical.

Vous faites d'abord glisser le composant ScrollPane du panneau Composants au panneau du document actuel puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant ScrollPane dans la bibliothèque
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";

// Défilement de 20 pixels lorsque l'utilisateur clique sur des flèches
// de la barre de défilement vertical.
my_sp.vLineScrollSize = 20;
```

ScrollPane.vPageScrollSize

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

scrollPaneInstance.vPageScrollSize

Description

Property : nombre indiquant de combien de pixels le contenu se déplace dans la zone d'affichage lorsque l'utilisateur clique sur le rail dans une barre de défilement verticale. La valeur par défaut est 20.

Exemple

Cet exemple crée une occurrence de composant ScrollPane appelée `my_sp`, la charge avec une image et définit la propriété `vPageScrollSize` pour faire défiler 30 pixels lorsque l'utilisateur clique à l'intérieur de la barre de défilement vertical.

Faites d'abord glisser le composant ScrollPane du panneau Composants jusqu'à la bibliothèque du document actif, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant ScrollPane dans la bibliothèque
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";

// Défilement de 30 pixels lorsque l'utilisateur clique sur la barre
// de défilement vertical.
my_sp.vPageScrollSize = 30;
```

ScrollPane.vPosition

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

scrollPaneInstance.vPosition

Description

Propriété ; oriente le contenu du panneau de défilement en pixels et ajuste le curseur de défilement vertical proportionnellement. La position 0 se trouve en haut du rail de défilement, ce qui permet de rendre visible le bord supérieur du contenu du panneau défilant dans ce dernier. La valeur par défaut est 0.

Exemple

Cet exemple crée une occurrence ScrollPane appelée `my_sp`, la charge avec une image et crée un écouteur pour gérer l'événement scroll et afficher les positions de défilement horizontal (`hPosition`) et vertical (`vPosition`) lorsque l'utilisateur clique sur la barre de défilement.

Faites d'abord glisser le composant ScrollPane du panneau Composants jusqu'à la bibliothèque du document actif, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant ScrollPane dans la bibliothèque
*/

this.createClassObject(mx.containers.ScrollPane, "my_sp", 10);
my_sp.setSize(360, 280);

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
// Défilement de 100 pixels lorsque l'utilisateur clique sur la barre
// horizontale.
my_sp.hPageScrollSize = 100;

// Création d'un objet écouteur.
var spListener:Object = new Object();
spListener.scroll = function(evt_obj:Object):Void {
   trace("hPosition = " + my_sp.hPosition + ", vPosition = " +
  my_sp.vPosition);
};
// Ajout de l'écouteur.
my_sp.addEventListener("scroll", spListener);
```

ScrollPane.vScrollPolicy

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

scrollPaneInstance.vScrollPolicy

Description

Propriété : détermine si la barre de défilement verticale est toujours présente (`"on"`), jamais présente (`"off"`) ou s'affiche automatiquement en fonction de la taille de l'image (`"auto"`). La valeur par défaut est `"auto"`.

Exemple

L'exemple suivant crée une occurrence d'un composant ScrollPane appelée `my_sp`, définit `vScrollPolicy` sur `off` pour que la barre de défilement vertical n'apparaisse pas, et charge le composant ScrollPane avec une image.

Faites d'abord glisser le composant ScrollPane du panneau Composants jusqu'à la bibliothèque du document en cours, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant ScrollPane dans la bibliothèque
*/
import mx.containers.ScrollPane;

this.createClassObject(ScrollPane, "my_sp", 30);
my_sp.setSize(360, 280);
my_sp.vScrollPolicy = "off";

System.security.allowDomain("http://www.helpexamples.com");
my_sp.contentPath = "http://www.helpexamples.com/flash/images/image1.jpg";
```

CHAPITRE 41

Classe SimpleButton

41

Héritage MovieClip > Classe UIObject > Classe UIComponent > SimpleButton

Nom de classe ActionScript mx.controls.SimpleButton

Les propriétés de la classe SimpleButton permettent de contrôler les éléments suivants lors de l'exécution :

- Si un bouton a l'aspect d'un bouton-poussoir par défaut
- Si le bouton agit comme un bouton-poussoir ou un bouton bascule
- Si un bouton est sélectionné

REMARQUE

La classe SimpleButton est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Méthodes de la classe SimpleButton

La classe SimpleButton ne présente pas de méthodes exclusives.

Méthodes héritées de la classe UIObject

Le tableau suivant répertorie les méthodes de la classe SimpleButton héritées de la classe UIObject. Pour appeler ces méthodes à partir de l'objet SimpleButton, utilisez le formulaire *buttonInstance.methodName*.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.

Méthode	Description
UIObject.getStyle()	Obtient la propriété de style de l'objet ou de la déclaration de style.
UIObject.invalidate()	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
UIObject.move()	Déplace l'objet à l'emplacement demandé.
UIObject.redraw()	Force la validation de l'objet pour qu'il soit dessiné dans l'image active.
UIObject.setSize()	Redimensionne l'objet à la taille demandée.
UIObject.setSkin()	Définit une enveloppe dans l'objet.
UIObject.setStyle()	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant répertorie les méthodes de la classe SimpleButton héritées de la classe UIComponent. Pour appeler ces méthodes à partir de l'objet SimpleButton, utilisez le formulaire *buttonInstance.methodName*.

Méthode	Description
UIComponent.getFocus()	Renvoie une référence à l'objet ayant le focus.
UIComponent.setFocus()	Attribue le focus à l'occurrence de composant.

Propriétés de la classe SimpleButton

Le tableau suivant répertorie les propriétés de la classe SimpleButton.

Propriété	Description
SimpleButton.emphasized	Indique si un bouton a l'aspect d'un bouton-poussoir par défaut.
SimpleButton.emphasizedStyleDeclaration	Déclaration de style lorsque la propriété `emphasized` est définie sur `true`.
SimpleButton.selected	Valeur booléenne indiquant si le bouton est sélectionné (`true`) ou non (`false`). La valeur par défaut est `false`.
SimpleButton.toggle	Valeur booléenne indiquant si le comportement du bouton est celui d'un bouton bascule (`true`) ou non (`false`). La valeur par défaut est `false`.

Propriétés héritées de la classe UIObject

Le tableau suivant répertorie les propriétés de la classe SimpleButton héritées de la classe UIObject. Pour accéder à ces propriétés à partir de l'objet SimpleButton, utilisez le formulaire *buttonInstance.propertyName*.

Propriété	Description
UIObject.bottom	Position du bord inférieur de l'objet par rapport au bord inférieur de son parent. Lecture seule.
UIObject.height	Hauteur de l'objet, en pixels. Lecture seule.
UIObject.left	Bord gauche de l'objet, en pixels. Lecture seule.
UIObject.right	Position du bord droit de l'objet par rapport au bord droit de son parent. Lecture seule.
UIObject.scaleX	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
UIObject.scaleY	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
UIObject.top	Position du bord supérieur de l'objet par rapport à son parent. Lecture seule.
UIObject.visible	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
UIObject.width	Largeur de l'objet, en pixels. Lecture seule.
UIObject.x	Bord gauche de l'objet, en pixels. Lecture seule.
UIObject.y	Bord supérieur de l'objet, en pixels. Lecture seule.

Propriétés héritées de la classe UIComponent

Le tableau suivant répertorie les propriétés de la classe SimpleButton héritées de la classe UIComponent. Pour accéder à ces propriétés à partir de l'objet SimpleButton, utilisez le formulaire *buttonInstance.propertyName*.

Propriété	Description
UIComponent.enabled	Indique si le composant peut recevoir le focus et la saisie.
UIComponent.tabIndex	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Evénements de la classe SimpleButton

Le tableau suivant présente l'événement de la classe SimpleButton.

Evénement	Description
SimpleButton.click	Diffusé lorsque l'utilisateur clique sur un bouton.

Evénements hérités de la classe UIObject

Le tableau suivant répertorie les événements de la classe SimpleButton hérités de la classe UIObject.

Evénement	Description
UIObject.draw	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
UIObject.hide	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
UIObject.load	Diffusé lorsque des sous-objets sont créés.
UIObject.move	Diffusé lorsque l'objet a été déplacé.
UIObject.resize	Diffusé lorsqu'un objet a été redimensionné.
UIObject.reveal	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
UIObject.unload	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant répertorie les événements de la classe SimpleButton hérités de la classe UIComponent.

Evénement	Description
UIComponent.focusIn	Diffusé lorsqu'un objet reçoit le focus.
UIComponent.focusOut	Diffusé lorsqu'un objet perd le focus.
UIComponent.keyDown	Diffusé lorsqu'une touche est enfoncée.
UIComponent.keyUp	Diffusé lorsqu'une touche est relâchée.

SimpleButton.click

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.click = function(eventObj:Object){
  // ...
};
buttonInstance.addEventListener("click", listenerObject);
```

Utilisation 2 :

```
on (click) {
  // ...
}
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque l'utilisateur clique sur le bouton (puis relâche le bouton de la souris) ou si le bouton a le focus et que l'utilisateur appuie sur la barre d'espacement.

Le premier exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*buttonInstance*) distribue un événement (ici, `click`) qui est géré par une fonction (également appelée *gestionnaire*) sur un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute. La méthode est appelée lorsque l'événement a lieu. Lorsque l'événement a lieu, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. L'objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `addEventListener()` (voir `EventDispatcher.addEventListener()`) sur l'occurrence de composant qui diffuse l'événement afin d'enregistrer l'écouteur avec cette occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence du composant Button. Le mot-clé `this`, utilisé dans un gestionnaire `on()` relié à un composant, fait référence à l'occurrence de ce composant. Par exemple, le code suivant, associé à l'occurrence Button `myButtonComponent`, envoie « _level0.myButtonComponent » vers le panneau Sortie :

```
on (click) {
  trace(this);
}
```

Le comportement de `this` est différent en cas d'utilisation dans un gestionnaire `on()` associé à un symbole de bouton Flash standard. Dans cet exemple, `this` fait référence au symbole que contient le bouton. Par exemple, le code suivant, lié à l'occurrence de symbole de bouton `myButton`, envoie « _level0 » vers le panneau Sortie :

```
on (release) {
  trace(this);
}
```

REMARQUE

L'objet bouton ActionScript intégré n'a pas d'événement `click` ; l'événement le plus proche est `release`.

Exemple

Cet exemple, écrit sur une image du scénario, envoie un message vers le panneau Sortie lorsque l'utilisateur clique sur un bouton intitulé `buttonInstance`. La première ligne lui applique un comportement de bouton bascule. La deuxième ligne crée un objet écouteur intitulé `form`. La troisième ligne définit une fonction pour l'événement `click` sur l'objet écouteur. La fonction comporte une instruction `trace()` qui utilise l'objet événement (ici `eventObj`) qui lui est automatiquement transmis pour générer un message. La propriété `target` d'un objet événement est le composant ayant généré l'événement (dans cet exemple, `buttonInstance`). L'utilisateur accède à la propriété `SimpleButton.selected` à partir de la propriété `target` de l'objet événement. La dernière ligne appelle la méthode `addEventListener()` à partir de `buttonInstance` et lui transmet l'événement `click` et l'objet écouteur `form` comme paramètres.

```
buttonInstance.toggle = true;
var form:Object = new Object();
form.click = function(eventObj:Object) {
  trace("The selected property has changed to " +
  eventObj.target.selected);
};
buttonInstance.addEventListener("click", form);
```

Le code suivant envoie également un message vers le panneau Sortie lorsque l'utilisateur clique sur `buttonInstance`. Le gestionnaire `on()` doit être directement lié à `buttonInstance`.

```
on (click) {
  trace("button component was clicked");
}
```

SimpleButton.emphasized

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`buttonInstance``.emphasized`

Description

Propriété : indique si le bouton est dans un état emphasized (`true`) ou non (`false`). L'état emphasized correspond à l'aspect d'un bouton-poussoir par défaut. En général, utilisez la propriété `FocusManager.defaultPushButton` au lieu de définir la propriété `emphasized` directement. La valeur par défaut est `false`.

Si vous n'utilisez pas `FocusManager.defaultPushButton`, il est possible que vous souhaitiez simplement définir un bouton sur l'état emphasized ou utiliser l'état emphasized pour choisir une autre couleur pour le texte. L'exemple suivant définit la propriété `emphasized` pour l'occurrence de bouton `myButton` :

```
_global.styles.foo = new CSSStyleDeclaration();
_global.styles.foo.color = 0xFF0000;
SimpleButton.emphasizedStyleDeclaration = "neutralStyle";
myButton.emphasized = true;
```

Voir aussi

`SimpleButton.emphasizedStyleDeclaration`

SimpleButton.emphasizedStyleDeclaration

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

buttonInstance.emphasizedStyleDeclaraion

Description

Propriété (statique) : chaîne indiquant la déclaration de style qui met en forme un bouton lorsque la propriété `emphasized` est définie sur `true`.

La propriété `emphasizedStyleDeclaration` est une propriété statique de la classe SimpleButton. Vous devez donc y accéder directement à partir de SimpleButton plutôt que depuis un *buttonInstance*, de la manière suivante :

```
SimpleButton.emphasizedStyleDeclaration = "3dEmphStyle";
```

Voir aussi

`SimpleButton.emphasized`

SimpleButton.selected

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

buttonInstance.selected

Description

Propriété : valeur booléenne indiquant si le bouton est sélectionné (`true`) ou non (`false`). La valeur par défaut est `false`.

SimpleButton.toggle

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

buttonInstance.toggle

Description

Propriété : valeur booléenne indiquant si le bouton agit comme un bouton bascule (`true`) ou non (`false`). La valeur par défaut est `false`.

Si un bouton agit comme un bouton bascule, il reste enfoncé jusqu'à ce que vous cliquiez de nouveau dessus pour le relâcher.

CHAPITRE 42

Classe Slide

42

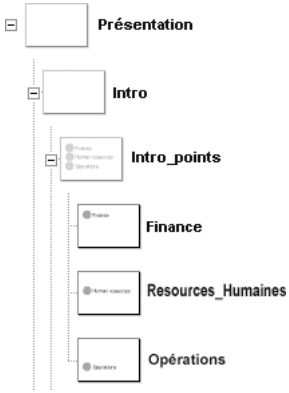
La classe Slide correspond à un noeud dans un diaporama hiérarchique que vous créez dans le panneau Contour de l'écran d'Adobe Flash CS3 Professional.

REMARQUE

La fonction Ecrans est délaissée dans Flash CS3. Vous pouvez ouvrir et modifier les fichiers FLA composés d'écrans qui ont été créés à partir des précédentes versions de Flash, mais vous ne pouvez plus créer ce type de document. Par ailleurs, notez que la classe Slide est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript2.0 dans ses paramètres de publication.

La classe Slide étend la classe Screen (voir « Classe Screen », à la page 1119), et fournit des fonctions intégrées de navigation et de séquencement entre les diapositives. Elle permet également d'ajouter facilement des transitions entre les diapositives à l'aide des comportements. Les objets Slide conservent une notion d'état de façon à ce que lorsqu'un utilisateur atteint la diapositive suivante, la diapositive précédente est masquée. Pour une présentation générale de l'utilisation des écrans, reportez-vous à *Utilisation des écrans* dans *Utilisation de Flash*.

Notez que vous pouvez uniquement accéder aux diapositives (vous « arrêter » sur les diapositives) ne contenant pas de diapositives enfant (« feuille »). L'illustration suivante affiche le contenu du panneau Contour de l'écran pour un exemple de diaporama :



Lorsque le diaporama commence, il « s'arrête » par défaut sur la diapositive appelée Finance, qui correspond à la première diapositive du diaporama ne contenant pas de diapositive enfant.

Notez que les diapositives enfant « héritent » de l'apparence visuelle (graphiques et autre contenu) de leurs diapositives parent. Par exemple, dans l'illustration ci-dessus, en plus du contenu de la diapositive Finance, l'utilisateur voit le contenu des diapositives Intro et Présentation.

REMARQUE

La classe Slide hérite de la Classe Loader qui permet de charger facilement des fichiers SWF ou JPEG externes dans une diapositive donnée. Cela permet de modulariser les diaporamas et de réduire le délai initial de téléchargement. Pour plus d'informations, voir « Chargement de contenu externe dans des écrans », à la page 1120.

Utilisation de la classe Slide

Utilisez les méthodes et propriétés de la classe Slide pour contrôler les diaporamas créés à l'aide du panneau Contour de l'écran pour un diaporama Flash, pour obtenir des informations sur un diaporama (par exemple, déterminer le nombre de diapositives enfant contenues dans la diapositive parent) ou pour naviguer entre les diapositives d'un diaporama (par exemple, pour créer les boutons « Diapositive suivante » et « Diapositive précédente »).

Vous pouvez également utiliser les comportements intégrés disponibles dans le panneau Comportements pour contrôler des diaporamas. Pour plus d'informations, reportez-vous à *Ajout de commandes sur les écrans à l'aide des comportements* dans *Utilisation de Flash*.

Paramètres de Slide

Vous pouvez définir les paramètres de création suivants pour chaque diapositive dans l'inspecteur Propriétés ou l'inspecteur de composants :

autoKeyNav détermine si la diapositive réagit aux interactions clavier par défaut et de quelle manière. Pour plus d'informations, voir `Slide.autoKeyNav`.

autoload indique si le contenu désigné par le paramètre contentPath doit être chargé automatiquement (`true`) ou attendre l'appel à la méthode `Loader.load()` (`false`). La valeur par défaut est `true`.

contentPath spécifie le contenu de la diapositive. Il peut s'agir de l'identificateur de liaison d'un clip ou d'une URL absolue ou relative d'un fichier SWF ou JPEG à charger dans la diapositive. Par défaut, le contenu chargé est coupé pour être adapté à la diapositive.

overlayChildren spécifie si les diapositives enfant de la diapositive restent visibles (`true`) ou non (`false`) lorsque vous passez d'une diapositive enfant à la suivante.

playHidden spécifie si la lecture de la diapositive continue lorsqu'elle est masquée (`true`) ou non (`false`).

Utilisation de la classe Slide pour créer un diaporama

Utilisez les méthodes et les propriétés de la classe Slide pour contrôler les diaporamas que vous créez dans le panneau Contour de l'écran pour un diaporama Flash de l'environnement de programmation de Flash. (Le panneau Comportements contient également plusieurs comportements permettant de créer la navigation entre des diapositives.) Dans cet exemple, vous écrivez votre propre code ActionScript pour créer les boutons Suivant et Précédent d'un diaporama.

Classe Slide (API)

Héritage MovieClip > Classe UIObject > Classe UIComponent > Affichage > Composant Loader > Classe Screen > Slide

Nom de classe ActionScript mx.screens.Slide

Les méthodes, propriétés et événements de la classe Slide vous permettent de gérer et de manipuler des diapositives.

Méthodes de la classe Slide

Le tableau suivant répertorie les méthodes de la classe Slide.

Méthode	Description
`Slide.getChildSlide()`	Renvoie la diapositive enfant spécifiée.
`Slide.gotoFirstSlide()`	Permet d'atteindre la première diapositive feuille dans la hiérarchie de sous-diapositives de la diapositive.
`Slide.gotoLastSlide()`	Permet d'atteindre la dernière diapositive feuille dans la hiérarchie de sous-diapositives de la diapositive.
`Slide.gotoNextSlide()`	Permet d'atteindre la diapositive suivante.
`Slide.gotoPreviousSlide()`	Permet d'atteindre la diapositive précédente.
`Slide.gotoSlide()`	Permet d'atteindre une diapositive spécifique.

Méthodes héritées de la classe UIObject

Le tableau suivant répertorie les méthodes de la classe Slide héritées de la classe UIObject. Pour appeler ces méthodes à partir de l'objet Slide, utilisez le formulaire *SlideInstance.methodName*.

Méthode	Description
UIObject.createClassObject()	Crée un objet dans la classe spécifiée.
UIObject.createObject()	Crée un sous-objet dans un objet.
UIObject.destroyObject()	Détruit une occurrence de composant.
UIObject.doLater()	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
UIObject.getStyle()	Obtient la propriété de style de l'objet ou de la déclaration de style.
UIObject.invalidate()	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
UIObject.move()	Déplace l'objet à l'emplacement demandé.
UIObject.redraw()	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
UIObject.setSize()	Redimensionne l'objet à la taille demandée.
UIObject.setSkin()	Définit une enveloppe dans l'objet.
UIObject.setStyle()	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant répertorie les méthodes de la classe Slide héritées de la classe UIComponent. Pour appeler ces méthodes à partir de l'objet Slide, utilisez le formulaire *SlideInstance.methodName*.

Méthode	Description
UIComponent.getFocus()	Renvoie une référence à l'objet ayant le focus.
UIComponent.setFocus()	Attribue le focus à l'occurrence de composant.

Méthodes héritées de la classe Loader

Le tableau suivant indique la méthode de la classe Slide héritée de la classe Loader. Pour appeler cette méthode à partir de l'objet Slide, utilisez le formulaire *`SlideInstance.methodName`*.

Méthode	Description
`Loader.load()`	Charge le contenu spécifié par la propriété `contentPath`.

Méthodes héritées de la classe Screen

Le tableau suivant présente la méthode de la classe Slide héritée de la classe Screen. Pour appeler cette méthode à partir de l'objet Slide, utilisez le formulaire *`SlideInstance.methodName`*.

Méthode	Description
`Screen.getChildScreen()`	Renvoie l'écran enfant de cet écran à un index spécifique.

Propriétés de la classe Slide

Le tableau suivant répertorie les propriétés de la classe Slide.

Propriété	Description
`Slide.autoKeyNav`	Détermine si la diapositive réagit aux interactions clavier par défaut pour naviguer vers la diapositive suivante ou précédente.
`Slide.currentChildSlide`	Lecture seule ; renvoie le premier enfant de la diapositive contenant la diapositive active.
`Slide.currentFocusedSlide`	Lecture seule ; renvoie la diapositive la plus éloignée dans l'arborescence (la diapositive la plus éloignée de la racine de l'arborescence des diapositives) et contenant le focus global actuel.
`Slide.currentSlide`	Lecture seule : renvoie la diapositive active.
`Slide.defaultKeydownHandler`	Fonction de rappel qui prend la priorité sur les interactions clavier par défaut pour la navigation entre les diapositives (flèches gauche et droite).
`Slide.firstSlide`	Lecture seule ; renvoie la première diapositive enfant de la diapositive n'ayant pas d'enfant.

Propriété	Description
`Slide.indexInParentSlide`	Lecture seule ; renvoie l'index de la diapositive (dont la numérotation commence à zéro) dans la liste des sous-diapositives de son parent.
`Slide.lastSlide`	Lecture seule ; renvoie la dernière diapositive enfant de la diapositive n'ayant pas d'enfant.
`Slide.nextSlide`	Lecture seule : renvoie la diapositive que vous atteindriez en appelant la méthode `mySlide.gotoNextSlide()` sans effectuer le déplacement.
`Slide.numChildSlides`	Lecture seule : renvoie le nombre de diapositives enfant de la diapositive.
`Slide.overlayChildren`	Détermine si les diapositives enfant de la diapositive sont visibles lorsque le contrôle passe d'une diapositive enfant à la suivante.
`Slide.parentIsSlide`	Lecture seule : renvoie une valeur booléenne indiquant si l'objet parent de la diapositive est également une diapositive (`true`) ou non (`false`).
`Slide.parentIsSlide`	Lecture seule ; diapositive contenant la diapositive active. Cette propriété peut être définie sur `null` s'il s'agit de la diapositive racine.
`Slide.playHidden`	Détermine si la lecture de la diapositive continue lorsqu'elle est masquée.
`Slide.previousSlide`	Lecture seule : renvoie la diapositive que vous atteindriez en appelant `mySlide.gotoPreviousSlide()` sans effectuer le déplacement.
`Slide.rootSlide`	Lecture seule : renvoie la racine de l'arborescence de diapositives contenant la diapositive.

Propriétés héritées de la classe UIObject

Le tableau suivant répertorie les propriétés de la classe Slide héritées de la classe UIObject. Pour accéder à ces propriétés à partir de l'objet Slide, utilisez le formulaire *`SlideInstance.propertyName`*.

Propriété	Description
`UIObject.bottom`	Position du bord inférieur de l'objet par rapport au bord inférieur de son parent. Lecture seule.
`UIObject.height`	Hauteur de l'objet, en pixels. Lecture seule.
`UIObject.left`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.right`	Position du bord droit de l'objet par rapport au bord droit de son parent. Lecture seule.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Position du bord supérieur de l'objet par rapport à son parent. Lecture seule.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Largeur de l'objet, en pixels. Lecture seule.
`UIObject.x`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.y`	Bord supérieur de l'objet, en pixels. Lecture seule.

Propriétés héritées de la classe UIComponent

Le tableau suivant répertorie les propriétés de la classe Slide héritées de la classe UIComponent. Pour accéder à ces propriétés à partir de l'objet Slide, utilisez le formulaire *`SlideInstance.propertyName`*.

Propriété	Description
`UIComponent.enabled`	Indique si le composant peut recevoir le focus et la saisie.
`UIComponent.tabIndex`	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Propriétés héritées de la classe Loader

Le tableau suivant répertorie les propriétés de la classe Slide héritées de la classe Loader. Pour accéder à ces propriétés à partir de l'objet Slide, utilisez le formulaire *SlideInstance.propertyName*.

Propriété	Description
Loader.autoLoad	Valeur booléenne indiquant si le contenu se charge automatiquement (true) ou si vous devez appeler Loader.load() (false).
Loader.bytesLoaded	Propriété en lecture seule indiquant le nombre d'octets ayant été chargés.
Loader.bytesTotal	Propriété en lecture seule indiquant le nombres d'octets du contenu.
Loader.content	Référence au contenu du chargeur. Cette propriété est en lecture seule.
Loader.contentPath	Chaîne indiquant l'URL du contenu devant être chargé.
Loader.percentLoaded	Nombre indiquant le pourcentage de contenu chargé. Cette propriété est en lecture seule.
Loader.scaleContent	Valeur booléenne indiquant si le contenu est dimensionné pour s'ajuster au chargeur (true) ou si le chargeur est dimensionné pour s'ajuster au contenu (false).

Propriétés héritées de la classe Screen

Le tableau suivant répertorie les propriétés de la classe Slide héritées de la classe Screen. Pour accéder à ces propriétés à partir de l'objet Slide, utilisez le formulaire *SlideInstance.propertyName*.

Propriété	Description
Screen.currentFocusedScreen	Lecture seule : renvoie l'écran contenant le focus global actuel.
Screen.indexInParent	Lecture seule ; renvoie l'index de l'écran (dont la numérotation commence à zéro) dans la liste des écrans enfant de son écran parent.
Screen.numChildScreens	Lecture seule : renvoie le nombre d'écrans enfant contenus dans l'écran.

Propriété	Description
`Screen.parentIsScreen`	Lecture seule : renvoie une valeur booléenne (`true` ou `false`) indiquant si l'objet parent de l'écran est lui-même un écran.
`Screen.rootScreen`	Lecture seule : renvoie l'écran racine de l'arborescence (ou sous-arborescence) contenant l'écran.

Evénements de la classe Slide

Le tableau suivant répertorie les événements de la classe Slide.

Evénement	Description
`Slide.hideChild`	Diffusé chaque fois qu'un enfant d'une diapositive passe de l'état visible à l'état invisible.
`Slide.revealChild`	Diffusé chaque fois qu'une diapositive enfant d'une diapositive passe de l'état invisible à l'état visible.

Evénements hérités de la classe UIObject

Le tableau suivant répertorie les événements de la classe Slide hérités de la classe UIObject.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant répertorie les événements de la classe Slide hérités de la classe UIComponent.

Evénement	Description
`UIComponent.focusIn`	Diffusé lorsqu'un objet reçoit le focus.
`UIComponent.focusOut`	Diffusé lorsqu'un objet perd le focus.
`UIComponent.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIComponent.keyUp`	Diffusé lorsqu'une touche est relâchée.

Evénements hérités de la classe Loader

Le tableau suivant répertorie les événements de la classe Slide hérités de la classe Loader.

Evénement	Description
`Loader.complete`	Déclenché à la fin du chargement du contenu.
`Loader.progress`	Déclenché pendant le chargement du contenu.

Evénements hérités de la classe Screen

Le tableau suivant répertorie les événements de la classe Slide hérités de la classe Screen.

Evénement	Description
`Screen.allTransitionsInDone`	Diffusé lorsque toutes les transitions « en entrée » appliquées à un écran sont terminées.
`Screen.allTransitionsOutDone`	Diffusé lorsque toutes les transitions « en sortie » appliquées à un écran sont terminées.
`Screen.mouseDown`	Diffusé lorsque l'utilisateur clique sur un objet (forme ou clip) appartenant directement à l'écran.
`Screen.mouseDownSomewhere`	Diffusé lorsque l'utilisateur clique quelque part sur la scène, mais pas nécessairement sur un objet appartenant à l'écran.
`Screen.mouseMove`	Diffusé lorsque le pointeur de la souris se déplace sur un écran.
`Screen.mouseOut`	Diffusé lorsque le pointeur de la souris se déplace de l'intérieur vers l'extérieur de l'écran.
`Screen.mouseOver`	Diffusé lorsque le pointeur de la souris se déplace de l'extérieur vers l'intérieur de l'écran.

Evénement	Description
`Screen.mouseUp`	Diffusé lorsque l'utilisateur relâche le bouton de la souris au-dessus d'un objet (forme ou clip) appartenant directement à l'écran.
`Screen.mouseUpSomewhere`	Diffusé lorsque l'utilisateur relâche le bouton de la souris quelque part sur la scène, mais pas nécessairement sur un objet appartenant à l'écran.

Slide.autoKeyNav

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

mySlide`.autoKeyNav`

Description

Propriété : détermine si la diapositive réagit aux interactions clavier par défaut pour naviguer vers la diapositive suivante ou précédente lorsque *mySlide* a le focus. Cette propriété accepte les valeurs de chaîne `"true"`, `"false"` et `"inherit"`. Vous pouvez modifier le comportement de traitement clavier par défaut à l'aide de la propriété `Slide.defaultKeydownHandler`.

Lorsque la valeur de cette propriété est définie sur `true`, une pression sur la flèche droite (`Key.RIGHT`) ou sur la barre d'espace (`Key.SPACE`) alors que *mySlide* a le focus permet d'accéder à la diapositive suivante ; une pression sur la flèche gauche (`Key.Left`) permet de revenir à la diapositive précédente.

Lorsque cette propriété est définie sur `false`, aucune interaction clavier par défaut ne se produit lorsque *mySlide* a le focus.

Lorsque cette propriété est définie sur `inherit`, *mySlide* examine la propriété `autoKeyNav` de sa diapositive parent. Si elle est également définie sur `inherit`, Flash tient compte de la chaîne d'héritage de la diapositive jusqu'à ce qu'elle trouve une diapositive parent dont la propriété `autoKeyNav` est définie sur `true` ou `false`.

Si *mySlide* n'a pas de diapositive parent (c'est-à-dire si l'instruction `(mySlide.parentIsSlide == false)` est `true`), elle se comporte comme si `autoKeyNav` avait été défini sur `true`.

Exemple

Cet exemple désactive la navigation clavier automatique pour la diapositive nommée `loginSlide`.

```
_root.Presentation.loginSlide.autoKeyNav = "false";
```

Voir aussi

`Slide.defaultKeydownHandler`

Slide.currentChildSlide

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`mySlide``.currentChildSlide`

Description

Propriété (lecture seule) : renvoie le premier enfant de *`mySlide`* contenant la diapositive active ; renvoie `null` si aucune diapositive enfant contenue dans *`mySlide`* n'a le focus actuel.

Exemple

Considérons par exemple le contour de l'écran suivant :

```
Presentation
  Slide_1
    Bullet1_1
      SubBullet1_1_1
    Bullet1_2
      SubBullet1_2_1
  Slide_2
```

Si `SubBullet1_1_1` est la diapositive active, toutes les instructions suivantes sont vraies :

```
Presentation.currentChildSlide == Slide_1;
Slide_1.currentChildSlide == Bullet_1_1;
SubBullet_1_1_1.currentChildSlide == null;
Slide_2.currentChildSlide == null;
```

Voir aussi

`Slide.currentSlide`

Slide.currentFocusedSlide

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
mx.screens.Slide.currentFocusedSlide
```

Description

Propriété (lecture seule) ; renvoie la diapositive la plus éloignée dans l'arborescence (la diapositive la plus éloignée de la racine de l'arborescence des diapositives) et contenant le focus global actuel. Le focus actuel peut se trouver dans la diapositive proprement dite, dans un clip, dans un objet texte ou dans un composant de la diapositive ; la méthode renvoie `null` s'il n'y a aucun focus actuel.

Exemple

```
var focusedSlide = mx.screens.Slide.currentFocusedSlide;
```

Slide.currentSlide

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
mySlide.currentSlide
```

Description

Propriété (lecture seule) : renvoie la diapositive active. Il s'agit toujours d'une diapositive « feuille », c'est-à-dire une diapositive ne contenant aucune diapositive enfant.

Exemple

Le code suivant, associé à un bouton de la diapositive du diaporama racine, permet de passer à la diapositive suivante du diaporama chaque fois que l'utilisateur clique sur le bouton.

```
// Associé à une occurrence de bouton contenue dans la diapositive du
// diaporama :
on(press) {
  _parent.currentSlide.gotoNextSlide();
}
```

Voir aussi

`Slide.gotoNextSlide()`

Slide.defaultKeydownHandler

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
mySlide.defaultKeyDownHandler = function (eventObj) {
  // Votre code ici.
}
```

Paramètres

`eventObj` Objet événement avec les propriétés suivantes :

- `type` Chaîne indiquant le type d'événement. Les valeurs possibles sont `"keyUp"` et `"keyDown"`.
- `ascii` Nombre entier représentant la valeur ASCII de la dernière touche enfoncée ; correspond à la valeur renvoyée par `Key.getAscii()`.
- `code` Nombre entier représentant le code de touche de la dernière touche enfoncée ; correspond à la valeur renvoyée par `Key.getCode()`.
- `shiftKey` Valeur booléenne indiquant si la touche Maj est enfoncée (`true`) ou non (`false`).
- `ctrlKey` Valeur booléenne indiquant si la touche Ctrl est enfoncée (`true`) ou non (`false`).

Renvoie

Aucune.

Description

Gestionnaire de rappel : permet de remplacer la navigation par défaut du clavier par un gestionnaire de clavier personnalisé que vous créez. Par exemple, plutôt que d'utiliser les flèches gauche et droite pour naviguer respectivement vers les diapositives précédente et suivante d'un diaporama, vous pouvez décider d'utiliser les flèches haut et bas. Pour plus d'informations sur le comportement de traitement des interactions clavier par défaut, reportez-vous à `Slide.autoKeyNav`.

Exemple

Dans cet exemple, le traitement des interactions clavier par défaut est modifié pour les diapositives enfant de la diapositive à laquelle le gestionnaire `on(load)` est associé. Ce gestionnaire utilise les flèches vers le haut et vers le bas pour la navigation au lieu des flèches vers la gauche et vers la droite.

```
on (load) {
  this.defaultKeyDownHandler = function(eventObj:Object) {
    switch (eventObj.code) {
    case Key.DOWN :
      this.currentSlide.gotoNextSlide();
      break;
    case Key.UP :
      this.currentSlide.gotoPreviousSlide();
      break;
    default :
      break;
    }
  };
}
```

Voir aussi

`Slide.autoKeyNav`

Slide.firstSlide

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

mySlide.firstSlide

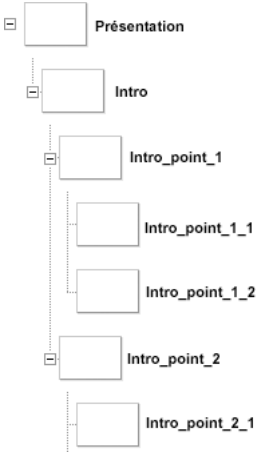
Description

Propriété (lecture seule) : renvoie la première diapositive enfant de *mySlide* n'ayant pas d'enfant.

Exemple

Dans la hiérarchie de diapositives illustrée ci-dessous, les deux instructions suivantes sont vraies :

```
Presentation.Intro.firstSlide == Intro_bullet_1_1;
Presentation.Intro_bullet_1.firstSlide == Intro_bullet_1_1;
```



Slide.getChildSlide()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

mySlide.getChildSlide(*childIndex*)

Paramètres

childIndex Index de la diapositive enfant à renvoyer ; la numérotation de l'index commence à zéro.

Renvoie

Un objet diapositive.

Description

Méthode : renvoie la diapositive enfant de *mySlide* dont l'index correspond à *childIndex*. Vous pouvez utiliser cette méthode pour itérer sur un jeu de diapositives enfant dont les index sont connus.

Exemple

Le code suivant affiche dans le panneau Sortie les noms de toutes les diapositives enfant de la diapositive du diaporama racine.

```
var numSlides = _root.Presentation.numChildSlides;
for(var slideIndex=0; slideIndex < numSlides; slideIndex++) {
  var childSlide = _root.Presentation.getChildSlide(slideIndex);
  trace(childSlide._name);
}
```

Voir aussi

Slide.numChildSlides

Slide.gotoFirstSlide()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
mySlide.gotoFirstSlide()
```

Paramètres

Aucun.

Renvoie

Aucune.

Description

Méthode : permet d'atteindre la première diapositive feuille dans l'arborescence de diapositives enfant sous *mySlide*. Cette méthode est ignorée lorsqu'elle est appelée depuis le gestionnaire d'événements `on(hide)` ou `on(reveal)` d'une diapositive, si l'événement résulte d'une opération de navigation dans les diapositives.

Pour atteindre la première diapositive d'un diaporama, appelez *mySlide*`.rootSlide.gotoFirstSlide()`. (Pour plus d'informations sur `rootSlide`, reportez-vous à `Slide.revealChild`.)

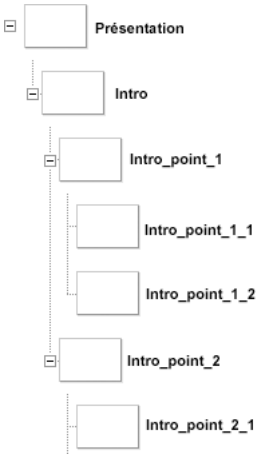
Exemple

Dans la hiérarchie de diapositives illustrée ci-dessous, les appels aux méthodes suivants permettent tous d'atteindre la diapositive nommée `Intro_bullet_1_1` :

```
Presentation.gotoFirstSlide();
Presentation.Intro.gotoFirstSlide();
Presentation.Intro.Intro_bullet_1.gotoFirstSlide();
```

Cette méthode permet d'atteindre la diapositive nommée `Intro_bullet_2_1` :

```
Presentation.Intro.Intro_bullet_2.gotoFirstSlide();
```



Voir aussi

`Slide.firstSlide`, `Slide.revealChild`

Slide.gotoLastSlide()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

mySlide`.gotoLastSlide()`

Paramètres

Aucun.

Renvoie

Aucune.

Description

Méthode : permet d'atteindre la dernière diapositive feuille dans l'arborescence de diapositives enfant sous *mySlide*. Cette méthode est ignorée lorsqu'elle est appelée depuis le gestionnaire d'événements `on(hide)` ou `on(reveal)` d'une diapositive, si l'événement résulte d'une opération de navigation dans les diapositives.

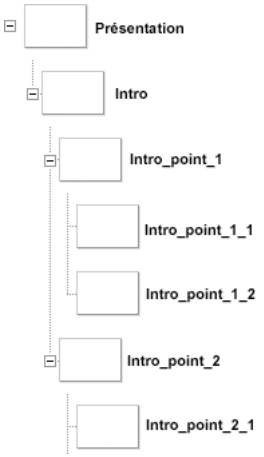
Exemple

Dans la hiérarchie de diapositives illustrée ci-dessous, les appels de méthode suivants permettent d'atteindre la diapositive nommée `Intro_bullet_1_2` :

```
Presentation.Intro.gotoLastSlide();
Presentation.Intro.Intro_bullet_1.gotoLastSlide();
```

Ces appels de méthode permettent d'atteindre la diapositive nommée `Intro_bullet_2_1` :

```
Presentation.gotoLastSlide();
Presentation.Intro.gotoLastSlide();
```



Voir aussi

`Slide.gotoSlide()`, `Slide.lastSlide`

Slide.gotoNextSlide()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

mySlide.gotoNextSlide()

Paramètres

Aucun.

Renvoie

Une valeur booléenne ou `null`. La méthode renvoie `true` si elle a permis d'atteindre la diapositive suivante ; elle renvoie `false` si la présentation se trouve déjà sur la dernière diapositive lorsque la méthode est invoquée (c'est-à-dire si `currentSlide.nextSlide` est `null`). La méthode renvoie `null` si elle est invoquée sur une diapositive ne contenant pas la diapositive actuelle.

Description

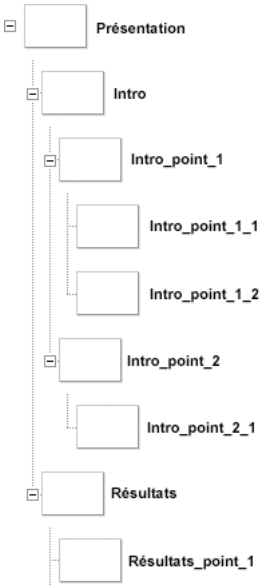
Méthode : permet d'atteindre la diapositive suivante du diaporama. Lorsque le contrôle passe d'une diapositive à la suivante, la diapositive sortante est masquée et la diapositive entrante est affichée. Si les diapositives se trouvent dans des sous-arborescences de diapositives différentes, toutes les diapositives ancêtre, depuis la diapositive sortante jusqu'à l'ancêtre commun des diapositives entrante et sortante, sont masquées et reçoivent un événement `hide`. Tout de suite après, toutes les diapositives ancêtre de la diapositive entrante, jusqu'à l'ancêtre commun des diapositives entrante et sortante, sont affichées et reçoivent un événement `reveal`.

La méthode `gotoNextSlide()` est généralement appelée sur le nœud feuille représentant la diapositive actuelle. Si elle est appelée sur un nœud non feuille, par exemple `someNode`, la méthode `someNode.gotoNextSlide()` permet d'atteindre le premier nœud feuille de la diapositive, ou « section. », suivante.

Cette méthode n'a aucun effet lorsqu'elle est appelée sur une diapositive qui ne contient pas la diapositive active. Cette méthode n'a pas plus d'effet lorsqu'elle est appelée depuis un gestionnaire d'événements `on(hide)` ou `on(reveal)` associé à une diapositive, si ce gestionnaire a été appelé suite à une opération de navigation dans les diapositives.

Exemple

Supposons que, dans la hiérarchie de diapositives suivante, la diapositive nommée `Intro_bullet_1_1` soit la diapositive en cours d'affichage (c'est-à-dire que `_root.Presentation.currentSlide._name == Intro_bullet_1_1`).



Dans ce cas, l'appel à la méthode `Intro_bullet_1_1.gotoNextSlide()` permet d'atteindre `Intro_bullet_1_2`, qui est un frère de `Intro_bullet_1_1`.

Cependant, l'appel à `Intro_bullet_1.gotoNextSlide()` permet d'atteindre `Intro_bullet_2_1`, la première diapositive feuille contenue dans `Intro_bullet_2`, qui est le frère suivant de `Intro_bullet_1`. De même, l'appel à `Intro.gotoNextSlide()` permet d'atteindre `Results_bullet_1`, la première diapositive feuille contenue dans la diapositive `Results`.

Enfin, en supposant toujours que la diapositive actuelle soit `Intro_bullet_1_1`, l'appel à `Results.gotoNextSlide()` n'a aucun effet, puisque `Results` ne contient pas la diapositive actuelle (ce qui signifie que `Results.currentSlide` est `null`).

Voir aussi

`Slide.currentSlide`, `Slide.gotoPreviousSlide()`, `Slide.nextSlide`

Slide.gotoPreviousSlide()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
mySlide.gotoPreviousSlide()
```

Paramètres

Aucun.

Renvoie

Une valeur booléenne ou `null`. La méthode renvoie `true` si elle a permis d'atteindre la diapositive précédente ; elle renvoie `false` si la présentation est à la première diapositive lorsque la méthode est invoquée (c'est-à-dire si `currentSlide.nextSlide` est `null`). La méthode renvoie `null` si elle est invoquée sur une diapositive ne contenant pas la diapositive actuelle.

Description

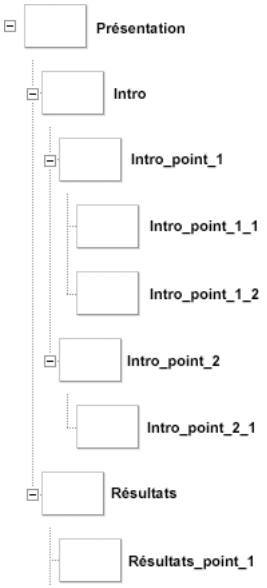
Méthode : permet d'atteindre la diapositive précédente du diaporama. Lorsque le contrôle passe d'une diapositive à la précédente, la diapositive sortante est masquée et la diapositive entrante est affichée. Si les diapositives se trouvent dans des sous-arborescences de diapositives différentes, toutes les diapositives ancêtre, depuis la diapositive sortante jusqu'à l'ancêtre commun des diapositives entrante et sortante, sont masquées et reçoivent un événement `hide`. Tout de suite après, toutes les diapositives ancêtre de la diapositive entrante, jusqu'à l'ancêtre commun des diapositives entrante et sortante, sont affichées et reçoivent un événement `reveal` .

La méthode `gotoPreviousSlide()` est généralement appelée sur le nœud feuille représentant la diapositive actuelle. Si elle est appelée sur un nœud non feuille, par exemple `someNode`, la méthode `someNode.gotoPreviousSlide()` permet d'atteindre le premier nœud feuille de la diapositive, ou « section. », précédente.

Cette méthode n'a aucun effet lorsqu'elle est appelée sur une diapositive qui ne contient pas la diapositive active. Cette méthode n'a pas plus d'effet lorsqu'elle est appelée depuis un gestionnaire d'événements `on(hide)` ou `on(reveal)` associé à une diapositive, si ce gestionnaire a été appelé suite à une opération de navigation dans les diapositives.

Exemple

Supposons que, dans la hiérarchie de diapositives suivante, la diapositive nommée `Intro_bullet_1_2` soit la diapositive en cours d'affichage (c'est-à-dire que `_root.Presentation.currentSlide._name == Intro_bullet_1_2`).



Dans ce cas, l'appel à la méthode `Intro_bullet_1_2.gotoPreviousSlide()` permet d'atteindre `Intro_bullet_1_1`, qui est le frère précédent de `Intro_bullet_1_2`.

Cependant, l'appel à `Intro_bullet_2.gotoPreviousSlide()` permet d'atteindre `Intro_bullet_1_1`, la première diapositive feuille contenue dans `Intro_bullet_1`, qui est le frère précédent de `Intro_bullet_2`. De même, l'appel à `Results.gotoPreviousSlide()` permet d'atteindre `Intro_bullet_1_1`, la première diapositive feuille contenue dans la diapositive `Intro`.

Enfin, si la diapositive actuelle est `Intro_bullet_1_1`, alors l'appel à `Results.gotoPreviousSlide()` n'a aucun effet, puisque `Results` ne contient pas la diapositive actuelle (ce qui signifie que `Results.currentSlide` est `null`).

Voir aussi

`Slide.currentSlide`, `Slide.gotoNextSlide()`, `Slide.previousSlide`

Slide.gotoSlide()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

mySlide.gotoSlide(*newSlide*)

Paramètres

newSlide Diapositive à atteindre.

Renvoie

Une valeur booléenne indiquant si la navigation a réussi (`true`) ou non (`false`).

Description

Méthode : permet d'atteindre la diapositive spécifiée par *newSlide*. Pour que la navigation réussisse, les conditions suivantes doivent être remplies :

- La diapositive courante doit être un enfant de *mySlide*.
- La diapositive spécifiée par *newSlide* et la diapositive courante doivent avoir un ancêtre de diapositive commun, ce qui signifie que la diapositive courante et *newSlide* doivent se trouver dans la même sous-arborescence de diapositives.

Si aucune de ces conditions n'est remplie, la navigation échoue et la méthode renvoie `false` ; sinon, la méthode atteint la diapositive spécifiée et renvoie `true`.

Considérons par exemple la hiérarchie de diapositives suivante :

```
Presentation
  Slide1
    Slide1_1
    Slide1_2
 Slide2
    Slide2_1
    Slide2_2
```

Si la diapositive actuelle est `Slide1_2`, alors l'appel à `gotoSlide()` suivant échoue, puisque la diapositive actuelle n'est pas un descendant de `Slide2` :

```
Slide2.gotoSlide(Slide2_1);
```

Considérons également la hiérarchie de diapositives suivante, dans laquelle un objet de formulaire est l'écran parent de deux arborescences de diapositives distinctes :

```
Form_1
  Slide1
    Slide1_1
    Slide1_2
  Slide2
    Slide2_1
    Slide2_2
```

Si la diapositive actuelle est `Slide1_2`, alors la méthode suivante échoue également, car `Slide1` et `Slide2` se situent dans des sous-arborescences de diapositives différentes :

```
Slide1_2.gotoSlide(Slide2_2);
```

Exemple

Le code suivant, associé à un composant Button, utilise la propriété `Slide.currentSlide` et la méthode `gotoSlide()` pour afficher la diapositive suivante dans le diaporama.

```
on(click) {
  _parent.gotoSlide(_parent.currentSlide.nextSlide);
}
```

C'est l'équivalent au code suivant qui utilise la méthode `Slide.gotoNextSlide()` :

```
on(click) {
  _parent.currentSlide.gotoNextSlide();
}
```

Voir aussi

`Slide.currentSlide`, `Slide.gotoNextSlide()`

Slide.hideChild

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
on(hideChild) {
  // Votre code ici.
}
```

Description

Evénement : diffusé chaque fois qu'un enfant d'une diapositive passe de l'état visible à l'état invisible. Cet événement est diffusé uniquement par diapositives, non par formulaires. L'événement `hideChild` sert essentiellement à appliquer des transitions « out » à tous les enfants d'une diapositive.

Exemple

Lorsqu'il est associé à la diapositive racine (par exemple, la diapositive du diaporama), ce code affiche le nom de chacune des diapositives enfant appartenant à la diapositive racine, au fur et à mesure qu'elles sont masquées.

```
on(hideChild) {
  var child = eventObj.target._name;
  trace(child + " has just been hidden");
}
```

Voir aussi

`Slide.revealChild`

Slide.indexInParentSlide

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`mySlide``.indexInParent`

Description

Propriété (lecture seule) ; renvoie l'index (basé sur zéro) de *`mySlide`* dans la liste des diapositives enfant de son parent.

Exemple

Le code suivant utilise les propriétés `indexInParentSlide` et `Slide.numChildSlides` pour afficher l'index de la diapositive en cours d'affichage ainsi que le nombre total de diapositives contenues dans la diapositive parent. Pour utiliser ce code, associez-le à une diapositive parent contenant une ou plusieurs diapositives enfant.

```
on (revealChild) {
  trace("Displaying "+(currentSlide.indexInParentSlide+1)+" of
  "+currentSlide._parent.numChildSlides);
}
```

Notez que, du fait que cette propriété est basée sur zéro, sa valeur est incrémentée de 1 (`currentSlide.indexInParent+1`) afin d'afficher des valeurs plus significatives.

Voir aussi

`Slide.numChildSlides`, `Slide.revealChild`

Slide.lastSlide

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`mySlide``.lastSlide`

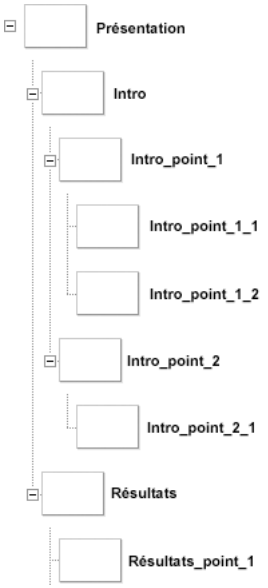
Description

Propriété (lecture seule) : renvoie la dernière diapositive enfant de *`mySlide`* n'ayant pas d'enfant.

Exemple

Toutes les instructions suivantes, concernant la hiérarchie de diapositives ci-dessous, sont vraies :

```
Presentation.lastSlide._name == Results_bullet_1;
Intro.lastSlide._name == Intro_bullet_1_2;
Intro_bullet_1.lastSlide._name == Intro_bullet_1_2;
Results.lastSlide._name = Results_bullet_1;
```



Slide.nextSlide

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

mySlide.nextSlide

Description

Propriété (lecture seule) : renvoie la diapositive que vous atteindriez en appelant *mySlide*.gotoNextSlide(), sans effectuer le déplacement. Vous pouvez par exemple utiliser cette propriété pour afficher le nom de la diapositive suivante du diaporama et permettre aux utilisateurs de décider de l'afficher ou non.

Exemple

Dans cet exemple, l'étiquette d'un composant Button nommé nextButton affiche le nom de la diapositive suivante du diaporama. S'il n'y a pas de diapositive suivante (si mySlide.nextSlide est null), l'étiquette du bouton est mise à jour pour indiquer à l'utilisateur qu'il a atteint la fin du diaporama.

```
if (mySlide.nextSlide != null) {
    nextButton.label = "Next slide: " + mySlide.nextSlide._name + " > ";
} else {
  nextButton.label = "End of this slide presentation.";
}
```

Voir aussi

Slide.gotoNextSlide(), Slide.previousSlide

Slide.numChildSlides

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

mySlide.numChildSlides

Description

Propriété (lecture seule) : renvoie le nombre de diapositives enfant contenues dans *mySlide*. Une diapositive peut contenir des formulaires ou d'autres diapositives ; si *mySlide* contient à la fois des diapositives et des formulaires, cette propriété renvoie uniquement le nombre de diapositives et ne compte pas les formulaires.

Exemple

Cet exemple utilise `Slide.numChildSlides` et la méthode `Slide.getChildSlide()` pour itérer sur toutes les diapositives enfant de la diapositive du diaporama racine. Il affiche ensuite leurs noms dans le panneau Sortie.

```
var numSlides = _root.Presentation.numChildSlides;
for(var slideIndex=0; slideIndex < numSlides; slideIndex++) {
  var childSlide = _root.Presentation.getChildSlide(slideIndex);
  trace(childSlide._name);
}
```

Voir aussi

`Slide.getChildSlide()`

Slide.overlayChildren

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`mySlide``.overlayChildren`

Description

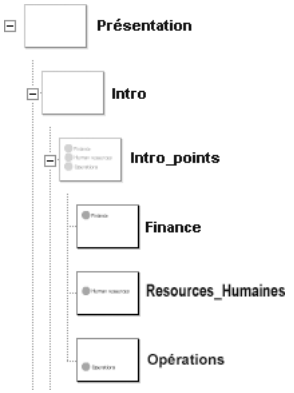
Propriété : détermine si les diapositives enfant de *`mySlide`* restent visibles lors de la navigation d'une diapositive enfant à la suivante. Lorsque cette propriété est définie sur `true`, la diapositive précédente reste visible lorsque le contrôle passe au frère suivant ; lorsqu'elle est définie sur `false`, la diapositive précédente est invisible lorsque le contrôle passe au frère suivant.

La définition de cette propriété sur `true` peut s'avérer utile, par exemple, lorsqu'une diapositive donnée contient plusieurs diapositives enfant de type « puce », révélées les unes après les autres (éventuellement à l'aide de transitions) et devant rester visibles lorsque les diapositives frère suivants s'affichent.

REMARQUE	Cette propriété s'applique uniquement aux descendants immédiats de *`mySlide`*, et non pas à toutes les diapositives enfant imbriquées.

Exemple

La diapositive Intro_bullets de l'illustration suivante contient trois diapositives enfant (Finance, Human_resources et Operations), affichant chacune une puce distincte. Si vous définissez `Intro_bullets.overlayChildren` sur `true`, toutes les diapositives de type puce restent sur la scène lorsque de nouvelles puces apparaissent.



Slide.parentIsSlide

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

mySlide`.parentIsSlide`

Description

Propriété (lecture seule) : valeur booléenne indiquant si l'objet parent de *mySlide* est également une diapositive. Si l'objet parent de *mySlide* est une diapositive ou qu'il appartient à une sous-classe de Slide, cette propriété renvoie `true` ; autrement, elle renvoie `false`.

Si *mySlide* est la diapositive racine d'un diaporama, cette propriété renvoie `false`, car le parent de la diapositive du diaporama est le scénario principal (_level0) et non une diapositive. Elle renvoie également `false` si le parent de *mySlide* est un formulaire.

Exemple

Le code suivant détermine si l'objet parent de la diapositive `mySlide` est lui-même une diapositive. Si `mySlide.parentIsSlide` est défini sur `true`, le nombre de frères de `mySlide` s'affiche dans le panneau Sortie. Si l'objet parent n'est pas une diapositive, Flash suppose que `mySlide` est la diapositive racine (maître) dans le diaporama et que par conséquent, elle n'a pas de frères.

```
if (mySlide.parentIsSlide) {
  trace("I have " + mySlide._parent.numChildSlides+" sibling slides");
} else {
  trace("I am the root slide and have no siblings");
}
```

Voir aussi

`Slide.numChildSlides`

Slide.parentIsSlide

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`mySlide``.parentSlide`

Description

Propriété (lecture seule) : référence à la diapositive qui contient la diapositive active.

Slide.playHidden

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

mySlide.playHidden

Description

Propriété : valeur booléenne spécifiant si la lecture de *mySlide* doit continuer lorsqu'elle est masquée. Lorsque cette propriété est définie sur `true`, la lecture de *mySlide* continue même lorsque celle-ci est masquée. Lorsque cette propriété est définie sur `false`, la lecture de *mySlide* s'arrête dès que celle-ci est masquée ; la lecture reprend à l'image 1 de *mySlide* dès que celle-ci est révélée de nouveau.

Slide.previousSlide

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

mySlide.previousSlide

Description

Propriété (lecture seule) : renvoie la diapositive que vous atteindriez en appelant *mySlide*.gotoPreviousSlide(), sans effectuer le déplacement. Vous pouvez par exemple utiliser cette propriété pour afficher le nom de la diapositive précédente du diaporama et permettre aux utilisateurs de décider de l'afficher ou non.

Exemple

Dans cet exemple, l'étiquette d'un composant Button nommé `previousButton` affiche le nom de la diapositive précédente du diaporama. S'il n'y a pas de diapositive précédente (si `mySlide.previousSlide` est `null`), l'étiquette du bouton est mise à jour pour indiquer à l'utilisateur qu'il se trouve au début du diaporama.

```
if (mySlide.previousSlide != null) {
    previousButton.label = "Previous slide: " + mySlide.previous._name + "
  > ";
} else {
    previousButton.label = "You're at the beginning of this slide
  presentation.";
}
```

Voir aussi

`Slide.gotoPreviousSlide()`, `Slide.nextSlide`

Slide.revealChild

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
on(revealChild) {
  // Votre code ici.
}
```

Description

Evénement : diffusé chaque fois qu'une diapositive enfant d'un objet diapositive passe de l'état invisible à l'état visible. Cet événement sert essentiellement à appliquer des transitions « en entrée » à toutes les diapositives enfant d'une diapositive donnée.

Exemple

Lorsqu'il est associé à la diapositive racine (la diapositive du diaporama, par exemple), ce code affiche le nom de chacune des diapositives enfant, au fur et à mesure de leur apparition.

```
on(revealChild) {
  var child = eventObj.target._name;
  trace(child + " has just appeared");
}
```

Voir aussi

`Slide.hideChild`

Slide.rootSlide

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

mySlide.rootSlide

Description

Propriété (lecture seule) : renvoie la diapositive racine de l'arborescence de diapositives ou de la sous-arborescence de diapositives contenant *mySlide*.

Exemple

Supposons que l'une de vos diapositives contienne un clip qui, lorsque l'utilisateur clique dessus, permet d'accéder à la première diapositive du diaporama. Pour obtenir ce résultat, vous devez associer le code suivant au clip :

```
on(press) {
  _parent.rootSlide.gotoFirstSlide();
}
```

Dans ce cas, _parent fait référence à la diapositive contenant l'objet clip.

CHAPITRE 43

Classe StyleManager

43

Nom de classe ActionScript mx.styles.StyleManager

La classe StyleManager conserve une trace des styles et des couleurs d'héritage connus. Vous avez besoin de cette classe uniquement si vous créez des composants et que vous souhaitez ajouter un nouveau style ou une nouvelle couleur d'héritage. Pour déterminer les styles d'héritage, accédez au site Web W3C à l'adresse www.w3.org/Style/CSS/.

REMARQUE	La classe StyleManager est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Méthodes de la classe StyleManager

Le tableau suivant répertorie les méthodes de la classe StyleManager.

Méthode	Description
`StyleManager.registerColorName()`	Enregistre un nouveau nom de couleur avec le gestionnaire de styles.
`StyleManager.registerColorStyle()`	Ajoute un nouveau style de couleur au gestionnaire de styles.
`StyleManager.registerInheritingStyle()`	Enregistre un nouveau style d'héritage avec le gestionnaire de styles.

StyleManager.registerColorName()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
StyleManager.registerColorName(colorName, value)
```

Paramètres

colorName Chaîne indiquant le nom de la couleur (par exemple, `"gray"`, `"darkGrey"`, etc.).

value Nombre hexadécimal indiquant la couleur (par exemple, 0x808080, 0x404040, etc.).

Valeur renvoyée

Aucune.

Description

Méthode ; associe un nom de couleur à une valeur hexadécimale et l'enregistre avec le gestionnaire de styles.

Exemple

Le code suivant enregistre `"gray"` comme nom de la couleur qui a pour valeur hexadécimale 0x808080 :

```
StyleManager.registerColorName("gray", 0x808080);
```

StyleManager.registerColorStyle()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
StyleManager.registerColorStyle(colorStyle)
```

Paramètres

colorStyle Chaîne indiquant le nom de la couleur (par exemple, « `highlightColor` », « `shadowColor` », « `disabledColor` », etc.).

Valeur renvoyée

Aucune.

Description

Méthode : ajoute un nouveau style de couleur au gestionnaire de styles.

Exemple

L'exemple suivant enregistre « `highlightColor` » comme un style de couleur :

```
StyleManager.registerColorStyle("highlightColor");
```

StyleManager.registerInheritingStyle()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
StyleManager.registerInheritingStyle(propertyName)
```

Paramètres

propertyName Chaîne indiquant le nom de la propriété du style (par exemple, « `newProp1` », « `newProp2` », etc.).

Valeur renvoyée

Aucune.

Description

Méthode : signale cette propriété de style comme héritage. Utilisez cette méthode pour enregistrer des propriétés de styles qui ne sont pas répertoriées dans la spécification CSS. N'utilisez pas cette méthode pour transformer des propriétés de style non-héritage en propriétés de style d'héritage.

Lorsque la valeur d'un style n'est pas héritée, vous pouvez définir son style uniquement sur une occurrence, pas sur une feuille de style personnalisée ni globale. Un style qui n'hérite pas de sa valeur est défini sur la feuille de style de classe et par conséquent, vous ne pouvez pas la définir sur une feuille de style globale ou personnalisée.

Exemple

L'exemple suivant enregistre `newProp1` comme un style d'héritage :

```
StyleManager.registerInheritingStyle("newProp1");
```

CHAPITRE 44

Classe SystemManager

44

Nom de classe ActionScript mx.managers.SystemManager

La classe SystemManager fonctionne automatiquement avec la classe FocusManager pour vous permettre de choisir la fenêtre de niveau supérieur qui est activée dans une application. Elle fournit également une propriété `screen` qui permet aux composants et aux clips d'accéder aux coordonnées de la scène.

REMARQUE

La classe SystemManager est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Propriétés de la classe SystemManager

Le tableau suivant présente la propriété de la classe SystemManager.

Propriété	Description
`SystemManager.screen`	Lecture seule : un objet contenant la taille et la position de la scène.

SystemManager.screen

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`SystemManager.screen`

Description

Propriété : objet avec des propriétés `x`, `y`, `width` et `height` qui indiquent la taille et la position de la scène.

Exemple

Si `Stage.align` n'est pas défini sur « `LT` », il est très difficile de savoir quelles coordonnées peuvent réellement être affichées.

Par exemple, supposons que vous souhaitiez placer un clip de filigrane dans le coin inférieur droit de la scène (semblable aux filigranes utilisés par de nombreuses chaînes de télévision). Le code suivant fonctionnerait dans tous les alignements de la scène pour une occurrence de clip `watermark`. Toutefois, un composant doit se trouver sur la scène ou dans la bibliothèque pour que SystemManager puisse exister dans votre clip :

```
import mx.managers.SystemManager;

var p1:Number = SystemManager.screen.width + SystemManager.screen.x -
  watermark._width;
var p2:Number = SystemManager.screen.height + SystemManager.screen.y -
  watermark._height;

watermark._x = p1;
watermark._y = p2;
```

CHAPITRE 45

Composant TextArea

45

Le composant TextArea renvoie l'objet TextField ActionScript natif à la ligne. Vous pouvez utiliser les styles pour personnaliser le composant TextArea. Lorsqu'une occurrence est désactivée, son contenu s'affiche dans une couleur représentée par le style disabledColor. Un composant TextArea peut également être formaté en HTML ou en tant que champ de mot de passe qui masque le texte. Reportez-vous à *Application d'une feuille de style à un composant TextArea* dans le guide *Formation à ActionScript 2.0 dans Adobe Flash*.

REMARQUE

Un composant TextArea est pris en charge pour ActionScript 2.0 et ActionScript 3.0. Ce document présente le composant de la version 2. Si vous utilisez celui de la version 3, reportez-vous à « Utilisation du composant TextArea » dans *Utilisation des composants ActionScript 3.0*.

Un composant TextArea peut être activé ou désactivé dans une application. Lorsqu'il est désactivé, il ne reçoit pas les informations en provenance de la souris ou du clavier. Lorsqu'il est activé, il suit les mêmes règles de focus, de sélection et de navigation qu'un objet TextField ActionScript. Lorsqu'une occurrence de TextArea a le focus, vous pouvez utiliser les touches suivantes pour le contrôler :

Touche	Description
Touches fléchées	Déplacent le point d'insertion d'une ligne vers le haut, le bas, la gauche ou la droite.
Pg. Suiv.	Effectue un déplacement d'un écran vers le bas.
Pg. Préc.	Effectue un déplacement d'un écran vers le haut.
Maj+Tab	Place le focus sur l'objet précédent.
Tab	Place le focus sur l'objet suivant.

Pour plus d'informations sur le contrôle du focus, reportez-vous à *Création de la navigation personnalisée du focus* dans *Utilisation des composants ActionScript 2.0* ou « Classe FocusManager », à la page 745.

Un aperçu en direct de chaque occurrence de TextArea reflète les modifications effectuées sur les paramètres dans l'inspecteur Propriétés ou l'inspecteur de composants pendant la programmation. Si une barre de défilement s'avère nécessaire, elle apparaît lors d'un aperçu en direct, mais ne fonctionnera pas. Lors d'un aperçu en direct, il n'est pas possible de sélectionner du texte et vous ne pouvez pas entrer de texte dans l'occurrence du composant sur la scène.

Lorsque vous ajoutez le composant TextArea à une application, vous pouvez utiliser le panneau Accessibilité pour le rendre accessible aux lecteurs d'écran.

Utilisation du composant TextArea

Vous pouvez utiliser un composant TextArea partout où vous avez besoin d'un champ de texte multiligne. Si vous avez besoin d'un champ de texte à ligne unique, utilisez le Composant TextInput. Par exemple, vous pouvez utiliser le composant TextArea comme un champ de commentaires dans un formulaire. Vous pouvez définir un écouteur qui vérifie si le champ est vide lorsqu'un utilisateur sort du champ. Cet écouteur peut afficher un message d'erreur indiquant qu'un commentaire doit être entré dans ce champ.

Paramètres de TextArea

Dans l'inspecteur Propriétés ou l'inspecteur de composants (option de menu Fenêtre > Inspecteur de composants), vous pouvez définir les paramètres de création suivants pour chaque occurrence TextArea :

editable indique si le composant TextArea est modifiable (`true`) ou non (`false`). La valeur par défaut est `true`.

html indique si le texte est mis au format HTML (`true`) ou non (`false`). Si HTML est défini sur `true`, vous pouvez mettre en forme le texte en utilisant la balise font. La valeur par défaut est `false`.

text indique le contenu du composant TextArea. Vous ne pouvez pas entrer de retour chariot dans l'inspecteur Propriétés ou l'inspecteur des composants. La valeur par défaut est "" (une chaîne vide).

wordWrap indique si le texte est renvoyé à la ligne automatiquement (`true`) ou non (`false`). La valeur par défaut est `true`.

REMARQUE	Si vous créez une occurrence TextArea à l'aide de la méthode `createClassObject()`, la valeur par défaut pour wordWrap est `false`.

Dans l'inspecteur des composants, vous pouvez définir les paramètres supplémentaires suivants pour chaque occurrence du composant TextArea (Fenêtre > Inspecteur de composants) :

maxChars est le nombre maximal de caractères que la zone de texte peut contenir. La valeur par défaut est `null` (c'est-à-dire illimitée).

restrict indique le jeu de caractères qu'un utilisateur peut entrer dans la zone de texte. La valeur par défaut est `undefined`. Voir « TextArea.restrict », à la page 1248.

enabled est une valeur booléenne qui indique si le composant peut recevoir le focus et l'entrée. La valeur par défaut est `true`.

password est une valeur booléenne qui indique si l'entrée est un mot de passe ou d'autre texte à masquer lorsqu'il est tapé. Flash masque les caractères d'entrée avec des astérisques. La valeur par défaut est `false`.

visible est une valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`). La valeur par défaut est `true`.

REMARQUE	Les propriétés minHeight et minWidth sont utilisées par des routines de dimensionnement internes. Elles sont définies dans UIObject et remplacées par différents composants selon vos besoins. Ces propriétés peuvent être utilisées pour créer un gestionnaire de dispositions personnalisées. Dans le cas contraire, la définition de ces propriétés dans l'inspecteur de composants n'a aucun effet visible.

Vous pouvez contrôler ces options et d'autres options du composant TextArea à l'aide des propriétés, méthodes et événements d'ActionScript. Pour plus d'informations, voir « Classe TextArea », à la page 1232.

Création d'une application avec le composant TextArea

La procédure suivante explique comment ajouter un composant TextArea à une application pendant la programmation. L'exemple configure un gestionnaire d'événements `focusOut` sur l'occurrence TextArea qui vérifie que l'utilisateur a tapé des données dans la zone de texte avant de donner le focus à une autre partie de l'interface.

Pour créer une application avec le composant TextArea :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser un composant TextArea du panneau Composants vers la scène et nommez l'occurrence **my_ta**.
3. Sélectionnez l'image 1 dans le scénario, ouvrez le panneau Actions et saisissez le code suivant :

```
/**
 Requiert :
  - Occurrence TextArea sur la scène (nom d'occurrence : my_ta)
*/

var my_ta:mx.controls.TextArea;

var taListener:Object = new Object();
taListener.focusOut = function(evt_obj:Object) {
 if (my_ta.length < 1) {
  trace("Please enter a comment");
 }
};
my_ta.addEventListener("focusOut", taListener);
```

   Ce code configure un gestionnaire d'événements `focusOut` sur l'occurrence TextArea qui vérifie que l'utilisateur a bien tapé quelque chose dans la zone de texte.

Vous pouvez obtenir la valeur du texte entré dans l'occurrence de composant TextArea, comme suit :

```
var ta_text:String = my_ta.text;
```

Personnalisation du composant TextArea

Vous pouvez transformer un composant TextArea horizontalement et verticalement, pendant la programmation et lors de l'exécution. Lors de la programmation, sélectionnez le composant sur la scène et utilisez l'outil Transformer librement ou une commande Modification > Transformer. Lors de l'exécution, utilisez `UIObject.setSize()` ou toute propriété ou méthode applicable de la Classe TextArea.

Lorsqu'un composant TextArea est redimensionné, la bordure est redimensionnée en fonction du nouveau cadre de sélection. Le cas échéant, les barres de défilement s'affichent sur les bords inférieur et droit du cadre. La zone de texte est alors redimensionnée dans la zone restante. Dans un composant TextArea, les éléments n'ont pas de taille fixe. Si le composant TextArea est trop petit pour afficher le texte, ce dernier est rogné.

Utilisation de styles avec le composant TextArea

Les propriétés de style `backgroundColor` et `borderStyle` du composant TextArea sont définies dans une déclaration de style de classe. Les styles de classe remplacent les styles globaux. Pour définir les propriétés de style `backgroundColor` et `borderStyle`, vous devez donc créer une autre déclaration de style personnalisée sur l'occurrence.

Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, reportez-vous à *Utilisation de styles pour personnaliser la couleur et le texte des composants* dans *Utilisation des composants ActionScript 2.0*.

Un composant TextArea prend en charge les styles suivants :

Style	Thème	Description
`backgroundColor`	Les deux	Couleur d'arrière-plan. La couleur par défaut est le blanc.
`borderStyle`	Les deux	Le composant TextArea utilise une occurrence de RectBorder en tant que bordure et répond aux styles définis pour cette classe. Voir « Classe RectBorder », à la page 1111. Le style de bordure par défaut est « `inset` ».
`marginLeft`	Les deux	Nombre indiquant la marge gauche pour le texte. La valeur par défaut est 0.
`marginRight`	Les deux	Nombre indiquant la marge droite pour le texte. La valeur par défaut est 0.

Style	Thème	Description
color	Les deux	Couleur du texte. La valeur par défaut est 0x0B333C pour le thème Halo et vide pour le thème Sample.
disabledColor	Les deux	Couleur du texte lorsque le composant est désactivé. La couleur par défaut est 0x848384 (gris foncé).
embedFonts	Les deux	Valeur booléenne qui indique si la police spécifiée dans fontFamily est de type intégré. Ce style doit être défini sur true si fontFamily fait référence à une police intégrée. Dans le cas contraire, la police intégrée n'est pas utilisée. Si ce style est défini sur true et que fontFamily ne fait pas référence à une police intégrée, aucun texte n'est affiché. La valeur par défaut est false.
fontFamily	Les deux	Nom de la police du texte. La valeur par défaut est "_sans".
fontSize	Les deux	Taille, en points, de la police. La valeur par défaut est 10.
fontStyle	Les deux	Style de police : "normal" ou "italic". La valeur par défaut est "normal".
fontWeight	Les deux	Epaisseur de la police : "none" ou "bold". La valeur par défaut est "none". Tous les composants peuvent également accepter la valeur "normal" au lieu de "none" pendant un appel à la méthode setStyle(), mais les appels suivants à getStyle() renvoient "none".
textAlign	Les deux	Alignement du texte : "left", "right", "center" ou "justify". (Le paramètre « justify » est pris en charge uniquement dans Flash Player 8). La valeur par défaut est "left".
textIndent	Les deux	Nombre indiquant le retrait du texte. La valeur par défaut est 0.
textDecoration	Les deux	Décoration du texte : "none" ou "underline". La valeur par défaut est "none".

Les composants TextArea et TextInput utilisent exactement les mêmes styles et sont souvent utilisés de la même façon. Par conséquent, ils partagent par défaut la même déclaration de style de niveau de classe.

Par exemple, le code suivant définit un style sur la déclaration TextInput mais il affecte à la fois les composants TextInput et TextArea.

```
_global.styles.TextInput.setStyle("disabledColor", 0xBBBBFF);
```

Pour séparer les composants et fournir des styles de niveau de classe pour l'un et non pour l'autre, créez une déclaration de style.

```
import mx.styles.CSSStyleDeclaration;
_global.styles.TextArea = new CSSStyleDeclaration();
_global.styles.TextArea.setStyle("disabledColor", 0xFFBBBB);
```

Cet exemple ne vérifie pas si `_global.styles.TextArea` existait avant d'être remplacé ; il est supposé que vous savez qu'il existe et que vous souhaitez le remplacer.

Vous pouvez rendre l'arrière-plan des composants TextArea transparent en définissant le style `backgroundColor` globalement sur une valeur `undefined`. Vous devez ensuite définir individuellement le style `backgroundColor` sur une couleur pour tous les composants TextArea qui ne doivent pas être transparents.

```
// Attribution d'arrière-plans transparents à tous les composants TextArea.
_global.styles.TextArea.backgroundColor = undefined;

// Affectation d'un arrière-plan blanc à cette occurrence de composant
// spécifique.
myTextArea2.setStyle( "backgroundColor", "white" );
```

Le composant TextArea prend en charge un jeu de styles de composant pour tout le texte du champ. Cependant, vous pouvez également afficher du code HTML compatible avec le rendu HTML de Flash Player. Pour afficher du texte HTML, définissez `TextArea.html` sur `true`.

Si vous définissez le composant TextArea pour qu'il affiche du texte HTML, le style du texte est défini à l'aide de la classe TextField.StyleSheet (consultez les détails de cette classe dans le *Guide de référence du langage ActionScript 2.0*). Voici un exemple d'utilisation :

1. Faites glisser un composant TextArea sur la scène et nommez l'occurrence `my_ta`.
2. Entrez ce code dans le panneau Actions pour l'image 1 du scénario :

```
var my_styles = new TextField.StyleSheet();
my_styles.setStyle("p", {fontFamily:'Arial,Helvetica,sans-serif',
  fontSize:'12px', color:'#CC6699'});
my_ta.styleSheet = my_styles;
my_ta.html = true;
my_ta.text = "<p>This is some text</p>";
```

Utilisation d'enveloppes avec le composant TextArea

Le composant TextArea utilise une occurrence de RectBorder pour sa bordure et des barres de défilement pour des images de défilement. Pour plus d'informations sur l'application d'enveloppe à ces éléments visuels, reportez-vous à « Classe RectBorder », à la page 1111 et à « Utilisation d'enveloppes avec le composant UIScrollBar », à la page 1448.

Classe TextArea

Héritage MovieClip > Classe UIObject > Classe UIComponent > View > ScrollView > TextArea

Nom de classe ActionScript mx.controls.TextArea

Les propriétés de la classe TextArea vous permettent de définir le contenu, le formatage et la position horizontale et verticale du texte lors de l'exécution. Vous pouvez également indiquer si le champ est modifiable et s'il s'agit d'un champ Mot de passe. Vous pouvez également limiter le nombre de caractères que l'utilisateur peut saisir.

La définition d'une propriété de la classe TextArea avec ActionScript annule le paramètre du même nom défini dans l'inspecteur Propriétés ou des composants.

Le composant TextArea annule le rectangle de focus par défaut de Flash Player et dessine un rectangle de focus personnalisé avec des angles arrondis.

Le composant TextArea prend en charge les styles CSS et tout style HTML pris en charge par Flash Player.

Toutes les classes de composants ont une propriété `version` qui correspond à une propriété de classe. Les propriétés de classe ne sont disponibles que pour la classe elle-même. La propriété `version` renvoie une chaîne qui indique la version du composant. Pour accéder à cette propriété, utilisez le code suivant :

```
trace(mx.controls.TextArea.version);
```

REMARQUE	Le code `trace(myTextAreaInstance.version);` renvoie `undefined`.

Méthodes de la classe TextArea

La classe TextArea ne présente pas de méthodes exclusives.

Méthodes héritées de la classe UIObject

Le tableau suivant répertorie les méthodes de la classe TextArea héritées de la classe UIObject. Pour appeler ces méthodes à partir de l'objet TextArea, utilisez le formulaire *TextAreaInstance.methodName*.

Méthode	Description
UIObject.createClassObject()	Crée un objet dans la classe spécifiée.
UIObject.createObject()	Crée un sous-objet dans un objet.
UIObject.destroyObject()	Détruit une occurrence de composant.
UIObject.doLater()	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
UIObject.getStyle()	Obtient la propriété de style de l'objet ou de la déclaration de style.
UIObject.invalidate()	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
UIObject.move()	Déplace l'objet à l'emplacement demandé.
UIObject.redraw()	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
UIObject.setSize()	Redimensionne l'objet à la taille demandée.
UIObject.setSkin()	Définit une enveloppe dans l'objet.
UIObject.setStyle()	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant répertorie les méthodes de la classe TextArea héritées de la classe UIComponent. Pour appeler ces méthodes à partir de l'objet TextArea, utilisez le formulaire *TextAreaInstance.methodName*.

Méthode	Description
UIComponent.getFocus()	Renvoie une référence à l'objet ayant le focus.
UIComponent.setFocus()	Attribue le focus à l'occurrence de composant.

Propriétés de la classe TextArea

Le tableau suivant répertorie les propriétés de la classe TextArea.

Propriété	Description
`TextArea.editable`	Valeur booléenne indiquant si le champ est modifiable (`true`) ou non (`false`).
`TextArea.hPosition`	Définit la position du texte horizontalement dans le champ.
`TextArea.hScrollPolicy`	Indique si la barre de défilement horizontale est toujours affichée (`"on"`), jamais affichée (`"off"`), ou affichée seulement lorsque nécessaire (`"auto"`).
`TextArea.html`	Valeur booléenne qui indique si la zone de texte peut être mise au format HTML.
`TextArea.length`	Lecture seule : nombre de caractères contenus dans la zone de texte.
`TextArea.maxChars`	Nombre maximal de caractères que la zone de texte peut contenir.
`TextArea.maxHPosition`	Lecture seule ; valeur maximale de `TextArea.hPosition`.
`TextArea.maxVPosition`	Lecture seule ; valeur maximale de `TextArea.vPosition`.
`TextArea.password`	Valeur booléenne indiquant si le champ est un champ de mot de passe (`true`) ou non (`false`).
`TextArea.restrict`	Jeu de caractères qu'un utilisateur peut entrer dans la zone de texte.
`TextArea.styleSheet`	Associe une feuille de style au composant TextArea spécifié.
`TextArea.text`	Contenu du texte d'un composant TextArea.
`TextArea.vPosition`	Nombre indiquant la position du défilement vertical.
`TextArea.vScrollPolicy`	Indique si la barre de défilement verticale est toujours affichée (`"on"`), jamais affichée (`"off"`), ou affichée seulement lorsque nécessaire (`"auto"`).
`TextArea.wordWrap`	Valeur booléenne indiquant si le texte est renvoyé à la ligne automatiquement (`true`) ou non (`false`).

Propriétés héritées de la classe UIObject

Le tableau suivant répertorie les propriétés de la classe TextArea héritées de la classe UIObject. Pour accéder à ces propriétés à partir de l'objet TextArea, utilisez le formulaire *TextAreaInstance.propertyName*.

Propriété	Description
UIObject.bottom	Lecture seule ; position du bord inférieur de l'objet par rapport au bord inférieur de son parent.
UIObject.height	Lecture seule ; hauteur de l'objet, en pixels.
UIObject.left	Lecture seule ; bord gauche de l'objet, en pixels.
UIObject.right	Lecture seule ; position du bord droit de l'objet par rapport au bord droit de son parent.
UIObject.scaleX	Lecture seule ; nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
UIObject.scaleY	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
UIObject.top	Lecture seule ; renvoie la position du bord supérieur de l'objet par rapport à son parent.
UIObject.visible	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
UIObject.width	Lecture seule ; largeur de l'objet, en pixels.
UIObject.x	Lecture seule ; bord gauche de l'objet, en pixels.
UIObject.y	Lecture seule ; bord supérieur de l'objet, en pixels.

Propriétés héritées de la classe UIComponent

Le tableau suivant répertorie les propriétés de la classe TextArea héritées de la classe UIComponent. Pour accéder à ces propriétés à partir de l'objet TextArea, utilisez le formulaire *TextAreaInstance.propertyName*.

Propriété	Description
UIComponent.enabled	Indique si le composant peut recevoir le focus et la saisie.
UIComponent.tabIndex	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Evénements de la classe TextArea

Le tableau suivant présente l'événement de la classe TextArea.

Evénement	Description
`TextArea.change`	Indique aux écouteurs que le texte a été modifié.
`TextArea.scroll`	Indique aux écouteurs que l'utilisateur a fait défiler du texte.

Evénements hérités de la classe UIObject

Le tableau suivant répertorie les événements de la classe TextArea hérités de la classe UIObject.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant répertorie les événements de la classe TextArea hérités de la classe UIComponent.

Evénement	Description
`UIComponent.focusIn`	Diffusé lorsqu'un objet reçoit le focus.
`UIComponent.focusOut`	Diffusé lorsqu'un objet perd le focus.
`UIComponent.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIComponent.keyUp`	Diffusé lorsqu'une touche est relâchée.

TextArea.change

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.change = function(eventObject:Object) {
  // ...
};
textAreaInstance.addEventListener("change", listenerObject);
```

Utilisation 2 :

```
on (change) {
  // ...
}
```

Description

Evénement : indique aux écouteurs que le texte a été modifié. Cet événement est diffusé après la modification du texte. Cet événement ne peut pas être utilisé pour empêcher l'ajout de certains caractères dans la zone de texte du composant. Utilisez plutôt `TextArea.restrict`.

Le premier exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`textAreaInstance`*) distribue un événement (ici, `change`) qui est géré par une fonction (également appelée *gestionnaire*) sur un objet écouteur (*`listenerObject`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute. La méthode est appelée lorsque l'événement est déclenché. Lors de son déclenchement, il transmet automatiquement un objet événement (*`eventObject`*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence TextArea. Le mot-clé `this`, utilisé dans un gestionnaire `on()` relié à un composant, fait référence à l'occurrence de ce composant. Par exemple, le code suivant, associé à l'occurrence `myTextArea`, envoie « _level0.myTextArea » vers le panneau Sortie :

```
on (change) {
  trace(this);
}
```

Exemple

Cet exemple utilise le modèle d'événement dispatcher/écouteur pour suivre le nombre total de modifications de la zone de texte dans un composant TextArea appelé `my_ta`.

Vous devez d'abord ajouter une occurrence du composant TextArea sur la scène et la nommer **my_ta** ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextArea sur la scène (nom d'occurrence : my_ta)
*/

var my_ta:mx.controls.TextArea;

// Création d'une variable Number pour suivre le nombre de changements
// apportés au composant TextArea.
var changeCount_num:Number = 0;

// Définition d'un objet écouteur.
var taListener:Object = new Object();
// Définition d'une fonction exécutée chaque fois que l'écouteur reçoit.
// Notification d'un changement dans le composant TextArea.
taListener.change = function(evt_obj:Object) {
 changeCount_num++;
 trace("Text has changed " + changeCount_num + " times now!");
 trace("It now contains: " + evt_obj.target.text);
 trace("");
};
// Enregistrement de l'objet écouteur avec l'occurrence du
// composant TextArea.
my_ta.addEventListener("change", taListener);
```

Voir aussi

`EventDispatcher.addEventListener()`

TextArea.editable

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textAreaInstance.editable

Description

Propriété : valeur booléenne qui indique si le composant est modifiable (`true`) ou non (`false`). La valeur par défaut est `true`.

Exemple

L'exemple suivant définit la propriété `editable` sur `false` pour empêcher l'utilisateur de modifier le texte chargé dans l'occurrence TextArea appelée `my_ta`.

Vous devez d'abord ajouter une occurrence du composant TextArea sur la scène et la nommer `my_ta` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextArea sur la scène (nom d'occurrence : my_ta)
*/

var my_ta:mx.controls.TextArea;

my_ta.setSize(320, 240);
my_ta.editable = false;

// Chargement du texte à afficher et définition du gestionnaire onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_ta.text = src;
    } else {
        my_ta.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

TextArea.hPosition

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textAreaInstance.hPosition

Description

Propriété ; définit la position (en pixels) du texte horizontalement dans le champ.
La valeur par défaut est 0.

Exemple

L'exemple suivant utilise un écouteur pour afficher la position horizontale actuelle dans le panneau Sortie lorsque l'utilisateur fait défiler vers l'avant et vers l'arrière le texte chargé dans l'occurrence TextArea appelée `my_ta`.

Vous devez d'abord ajouter une occurrence du composant TextArea sur la scène et la nommer `my_ta` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextArea sur la scène (nom d'occurrence : my_ta)
*/

var my_ta:mx.controls.TextArea;

my_ta.setSize(320, 240);
my_ta.wordWrap = false;

var taListener:Object = new Object();
taListener.scroll = function(evt_obj:Object) {
 trace("hPosition = " + my_ta.hPosition);
}
my_ta.addEventListener("scroll", taListener);

// Chargement du texte à afficher et définition du gestionnaire onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_ta.text = src;
  my_ta.hPosition = 200;
    } else {
        my_ta.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

TextArea.hScrollPolicy

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textAreaInstance.hScrollPolicy

Description

Propriété : détermine si la barre de défilement horizontale est toujours affichée « (on »), jamais affichée (« off ») ou si elle doit s'afficher automatiquement en fonction de la taille du champ (« auto »). La valeur par défaut est « auto ».

Exemple

L'exemple suivant désactive la propriété hScrollPolicy, ce qui masque la barre de défilement de l'occurrence TextArea.

Vous devez d'abord ajouter une occurrence du composant TextArea sur la scène et la nommer my_ta ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextArea sur la scène (nom d'occurrence : my_ta)
*/

var my_ta:mx.controls.TextArea;

my_ta.setSize(320, 240);
my_ta.wordWrap = false;
my_ta.hScrollPolicy = "off";

// Chargement du texte à afficher et définition du gestionnaire onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_ta.text = src;
    } else {
        my_ta.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

TextArea.html

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textAreaInstance.html

Description

Propriété : valeur booléenne indiquant si le contenu de la zone de texte peut être mis au format HTML (`true`) ou non (`false`). Si la propriété `html` est définie sur `true`, le contenu de la zone de texte est au format HTML. Si `html` est `false`, la zone de texte est dans un autre format. La valeur par défaut est `false`.

Exemple

L'exemple suivant transforme le composant TextArea `my_ta` en zone de texte HTML, puis met le texte en forme avec des balises HTML.

Vous devez d'abord ajouter une occurrence du composant TextArea sur la scène et la nommer **my_ta** ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextArea sur la scène (nom d'occurrence : my_ta)
*/

var my_ta:mx.controls.TextArea;

my_ta.setSize(320, 240);
my_ta.html = true;
my_ta.text = "The <b>Royal</b> Nonesuch";
```

TextArea.length

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textAreaInstance.length

Description

Propriété (lecture seule) : indique le nombre de caractères d'une zone de texte. Cette propriété renvoie la même valeur que la propriété `text.length` ActionScript, mais elle est plus rapide. Un caractère tel que tab (« `\t` ») compte comme un seul caractère. La valeur par défaut est 0.

Exemple

L'exemple suivant accède à la propriété length pour afficher le nombre de caractères tapés par l'utilisateur dans le composant TextArea appelé `my_ta`.

Vous devez d'abord ajouter une occurrence du composant TextArea sur la scène et la nommer `my_ta` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextArea sur la scène (nom d'occurrence : my_ta)
*/

var my_ta:mx.controls.TextArea;

// Définition d'un objet écouteur.
var taListener:Object = new Object();
taListener.change = function(evt_obj:Object) {
 trace("my_ta.length is now: " + my_ta.length + " characters");
};
my_ta.addEventListener("change", taListener);
```

TextArea.maxChars

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textAreaInstance.maxChars

Description

Propriété : nombre maximal de caractères qu'une zone de texte peut contenir. Un script peut insérer plus de texte que la propriété `maxChars` ne le permet ; la propriété n'indique que la quantité de texte qu'un utilisateur peut entrer. Si la valeur de cette propriété est `null`, il n'y a pas de limite à la quantité de texte qu'un utilisateur peut entrer. La valeur par défaut est `null`.

Exemple

L'exemple suivant définit la propriété `maxchars` pour limiter à 24 le nombre de caractères qu'un utilisateur peut saisir.

Vous devez d'abord ajouter une occurrence du composant TextArea sur la scène et la nommer **my_ta** ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextArea sur la scène (nom d'occurrence : my_ta)
*/

var my_ta:mx.controls.TextArea;

my_ta.maxChars = 24;

// Définition d'un objet écouteur.
var taListener:Object = new Object();
taListener.change = function(evt_obj:Object) {
 trace("my_ta.length is now: " + my_ta.length + " characters");
};
my_ta.addEventListener("change", taListener);
```

TextArea.maxHPosition

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textAreaInstance.maxHPosition

Description

Propriété en lecture seule : valeur maximale de TextArea.hPosition. La valeur par défaut est 0.

Exemple

L'exemple suivant accède à la propriété maxHPosition pour définir la position initiale dans le composant TextArea appelé my_ta sur la position la plus à droite.

Vous devez d'abord ajouter une occurrence du composant TextArea sur la scène et la nommer my_ta ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextArea sur la scène (nom d'occurrence : my_ta)
*/

var my_ta:mx.controls.TextArea;

my_ta.setSize(320, 240);
my_ta.wordWrap = false;

var taListener:Object = new Object();
taListener.scroll = function(evt_obj:Object) {
 trace("hPosition = " + my_ta.hPosition);
}
my_ta.addEventListener("scroll", taListener);

// Chargement du texte à afficher et définition du gestionnaire onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_ta.text = src;
  my_ta.hPosition = my_ta.maxHPosition;
    } else {
        my_ta.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

Voir aussi

TextArea.vPosition

TextArea.maxVPosition

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textAreaInstance.maxVPosition

Description

Propriété en lecture seule : indique la valeur maximale de TextArea.vPosition. La valeur par défaut est 0.

Exemple

L'exemple suivant accède à la propriété maxVPosition pour définir la position verticale initiale du composant TextArea appelé my_ta sur le bas. Il présente également la position verticale actuelle lorsque l'utilisateur utilise le défilement vers le haut et vers le bas.

Vous devez d'abord ajouter une occurrence du composant TextArea sur la scène et la nommer my_ta ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextArea sur la scène (nom d'occurrence : my_ta)
*/

var my_ta:mx.controls.TextArea;

my_ta.setSize(320, 240);
my_ta.wordWrap = true;

var taListener:Object = new Object();
taListener.scroll = function(evt_obj:Object) {
 trace("vPosition = " + my_ta.vPosition);
}
my_ta.addEventListener("scroll", taListener);

// Chargement du texte à afficher et définition du gestionnaire onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
```

```
    if (src != undefined) {
        my_ta.text = src;
     my_ta.vPosition = my_ta.maxVPosition;
    } else {
        my_ta.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

Voir aussi

TextArea.hPosition

TextArea.password

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textAreaInstance.password

Description

Propriété : valeur booléenne indiquant si la zone de texte est un champ de mot de passe (`true`) ou non (`false`). Si la valeur de `password` est `true`, la zone de texte est une zone de texte de mot de passe dont le contenu est masqué avec des astérisques. Si la valeur de `password` est `false`, la zone de texte n'est pas une zone de texte de mot de passe. La valeur par défaut est `false`.

Exemple

L'exemple suivant traite le texte du composant TextArea `my_ta` comme un champ de mot de passe si la case à cocher `my_ch` est activée. Autrement, il le traite comme du texte ordinaire.

Vous devez d'abord ajouter une occurrence du composant TextArea sur la scène et la nommer `my_ta`, puis ajouter une case à cocher et la nommer `my_ch` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextArea sur la scène (nom d'occurrence : my_ta)
  - occurrence du composant CheckBox sur la scène (nom d'occurrence : my_ch)
*/
var my_ta:mx.controls.TextArea;
```

```
var my_ch:mx.controls.CheckBox;

my_ta.wordWrap = false;
my_ta.password = true;
my_ch.selected = my_ta.password;

var chListener:Object = new Object();
chListener.click = function(evt_obj:Object) {
 my_ta.password = my_ch.selected;
}
my_ch.addEventListener("click", chListener);
```

TextArea.restrict

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textAreaInstance.restrict

Description

Propriété : indique le jeu de caractères qu'un utilisateur peut entrer dans la zone de texte. La valeur par défaut est `undefined`. Si cette propriété est `null`, l'utilisateur peut entrer n'importe quel caractère. Si cette propriété est une chaîne vide, aucun caractère ne peut être entré. Si cette propriété est une chaîne de caractères, l'utilisateur peut entrer uniquement les caractères de la chaîne ; la chaîne est lue de gauche à droite. Vous pouvez spécifier une plage en utilisant un tiret (-).

Si la chaîne commence par ^, tous les caractères qui suivent le ^ sont considérés comme inacceptables. Si la chaîne ne commence pas par ^, les caractères dans la chaîne sont considérés comme acceptables. Le ^ peut également permettre de basculer entre des caractères acceptables et inacceptables.

Par exemple, le code suivant permet d'entrer les caractères A à Z sauf X et Q :

```
Ta.restrict = "A-Z^XQ";
```

Lorsque vous limitez la saisie à des caractères majuscules, les caractères alphabétiques saisis en minuscules sont convertis en majuscules. et vice-versa.

La propriété `restrict` limite seulement l'interaction de l'utilisateur, un script pouvant mettre n'importe quel texte dans la zone de texte. Cette propriété ne se synchronise pas avec les cases à cocher de polices vectorielles intégrées de l'inspecteur Propriétés.

Exemple

L'exemple suivant définit d'abord la propriété restrict pour limiter la zone de texte aux lettres majuscules, chiffres et espaces, puis la définit de façon à autoriser tous les caractères sauf les lettres minuscules.

Vous devez d'abord ajouter une occurrence du composant TextArea sur la scène et la nommer **my_ta** ; ajoutez ensuite le code suivant à l'image 1. Utilisez un seul paramètre à la fois pour la propriété `restrict`.

```
/**
 Requiert :
  - Occurrence TextArea sur la scène (nom d'occurrence : my_ta)
*/
var my_ta:mx.controls.TextArea;

my_ta.wordWrap = true;

// Limitation du contrôle aux lettres majuscules, aux nombres et
// aux espaces.
my_ta.restrict = "A-Z 0-9";

// Autorisation de tous les caractères, à l'exception des lettres minuscules
// caractères sur majuscules
my_ta.restrict = "^a-z";
```

TextArea.scroll

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.scroll = function(eventObject:Object) {
  // ...
};
textAreaInstance.addEventListener("scroll", listenerObject);
```

Utilisation 2 :

```
on (scroll) {
  // ...
}
```

Description

Evénement ; diffusé à tous les écouteurs enregistrés lorsque l'utilisateur clique sur la barre de défilement (puis relâche le bouton de la souris). La propriété `UIScrollBar.scrollPosition` et l'image à l'écran de la barre de défilement sont mises à jour avant la diffusion de cet événement.

Le premier exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur dans lequel le script est placé sur une image dans le scénario qui contient l'occurrence de composant. Une occurrence de composant (*`textAreaInstance`*) distribue un événement (ici, `scroll`) qui est géré par une fonction (également appelée *gestionnaire*) sur un objet écouteur (*`listenerObject`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement a lieu. Lorsque l'événement a lieu, il transmet automatiquement un objet événement (*`eventObject`*) à la méthode d'objet écouteur. L'objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `addEventListener()` (voir `EventDispatcher.addEventListener()`) sur l'occurrence de composant qui diffuse l'événement afin d'enregistrer l'écouteur avec cette occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Outre les propriétés normales de l'objet événement (`type` et `target`), l'objet événement pour l'événement `scroll` inclut une troisième propriété appelée `direction`. La propriété `direction` contient une chaîne décrivant l'orientation de la barre de défilement. Les valeurs possibles de la propriété `direction` sont `vertical` (valeur par défaut) et `horizontal`.

Pour plus d'informations sur les propriétés de l'objet événement `type` et `target`, reportez-vous à « Objets événement », à la page 516.

Le second exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence du composant TextArea. Le mot-clé `this`, utilisé dans un gestionnaire `on()` relié à un composant, fait référence à l'occurrence de ce composant. Par exemple, le code suivant, associé à l'occurrence TextArea `myTextAreaComponent`, envoie « _level0.myTextAreaComponent » vers le panneau Sortie :

```
on (scroll) {
  trace(this);
}
```

Exemple

Cet exemple utilise le modèle d'événement dispatcher/écouteur pour savoir quand l'utilisateur fait défiler l'occurrence TextArea à l'aide de ses barres de défilement ou de sa case de défilement.

Vous devez d'abord ajouter une occurrence du composant TextArea sur la scène et la nommer **my_ta** ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextArea sur la scène (nom d'occurrence : my_ta)
*/

my_ta.setSize(320, 240);
my_ta.move(10, 10);

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function(src:String):Void {
  my_ta.text = src;
}
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");

my_ta.addEventListener("scroll", doScroll);
function doScroll(evt_obj:Object):Void {
  trace("target:    " + evt_obj.target);
  trace("type:      " + evt_obj.type);
  trace("direction: " + evt_obj.direction);
  trace("position:  " + evt_obj.position);
  trace("");
}
```

Voir aussi

`EventDispatcher.addEventListener()`

TextArea.styleSheet

Disponibilité

Flash Player 7.

Utilisation

`textAreaInstance``.styleSheet = `*`TextFieldStyleSheetObject`*

Description

Propriété : associe une feuille de style au composant TextArea spécifié par *`TextAreaInstance`*. Pour plus d'informations sur la création des feuilles de style, reportez-vous à *Formatage de texte avec les feuilles de style CSS* dans le guide *Formation à ActionScript 2.0 dans Adobe Flash*.

La feuille de style associée à un composant TextArea peut être modifiée à tout moment. Si la feuille de style en cours d'utilisation est modifiée, le composant TextArea est redessiné avec la nouvelle feuille de style. La feuille de style peut être définie sur `null` ou sur `undefined` pour la supprimer. Si la feuille de style en cours d'utilisation est supprimée, le composant TextArea est redessiné sans feuille de style. La mise en forme effectuée par une feuille de style n'est pas conservée si cette dernière est supprimée.

Exemple

Le code suivant crée un objet StyleSheet nommé `my_styles` avec le constructeur `new TextField.StyleSheet`. Il définit ensuite des styles pour les balises `html` et `body`. Ensuite, il applique le style en affectant `my_styles` à la propriété `styleSheet` de l'occurrence TextArea `my_ta`.

Vous devez d'abord ajouter une occurrence du composant TextArea sur la scène et la nommer `my_ta` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextArea sur la scène (nom d'occurrence : my_ta)
*/

var my_ta:mx.controls.TextArea;

my_ta.setSize(320, 240);

// Création de l'objet StyleSheet.
var my_styles:TextField.StyleSheet = new TextField.StyleSheet();
my_styles.setStyle("html", {fontFamily:"Arial,Helvetica,sans-serif",
  fontSize:"12px", color:"#0000FF"});
my_styles.setStyle("body", {color:"#00CCFF", textDecoration:"underline"});

// Définition de la propriété TextAreaInstance.styleSheet sur le nouvel
// objet styleSheet nommé styles.
my_ta.styleSheet = my_styles;
my_ta.html = true;

// Chargement du texte pour afficher et définir le gestionnaire onLoad.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
 if (src != undefined) {
  my_ta.text = src;
 } else {
  my_ta.text = "Error loading HTML document.";
 }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.html");
```

TextArea.text

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textAreaInstance.text

Description

Propriété : le contenu du texte d'un composant TextArea. La valeur par défaut est "" (une chaîne vide).

Exemple

L'exemple suivant place une chaîne dans la propriété `text` de l'occurrence `my_ta TextArea`, puis envoie cette chaîne vers le panneau Sortie.

Vous devez d'abord ajouter une occurrence du composant TextArea sur la scène et la nommer `my_ta` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextArea sur la scène (nom d'occurrence : my_ta)
*/

var my_ta:mx.controls.TextArea;

my_ta.text = "The Royal Nonesuch";
trace(my_ta.text); // Présentation de « The Royal Nonesuch ».
```

TextArea.vPosition

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textAreaInstance.vPosition

Description

Propriété : définit la position de défilement verticale du texte dans une zone de texte. Cette propriété est utile pour diriger les utilisateurs vers un paragraphe spécifique dans un long passage, ou pour créer des zones de texte à défilement. Vous pouvez obtenir et définir cette propriété. La valeur par défaut est 0.

Exemple

L'exemple suivant charge du texte dans le composant TextArea appelé `my_ta` et définit la propriété vPosition pour qu'elle affiche le texte en bas.

Vous devez d'abord ajouter une occurrence du composant TextArea sur la scène et la nommer `my_ta` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextArea sur la scène (nom d'occurrence : my_ta)
*/

var my_ta:mx.controls.TextArea;

my_ta.setSize(320, 240);

// Chargement du texte à afficher et définition du gestionnaire onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
 if (src != undefined) {
  my_ta.text = src;
  my_ta.vPosition = my_ta.maxVPosition;
 } else {
  trace("Error loading text.");
 }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt")
```

TextArea.vScrollPolicy

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textAreaInstance.vScrollPolicy

Description

Propriété : détermine si la barre de défilement verticale est toujours affichée (« on »), jamais affichée (« off »), ou si elle s'affiche automatiquement en fonction de la taille du champ (« auto »). La valeur par défaut est « auto ».

Exemple

L'exemple suivant désactive la barre de défilement vertical pour le composant TextArea appelé my_ta de façon à ce que la barre de défilement ne soit pas disponible pour faire défiler le texte chargé par l'exemple.

Vous devez d'abord ajouter une occurrence du composant TextArea sur la scène et la nommer my_ta ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextArea sur la scène (nom d'occurrence : my_ta)
*/
var my_ta:mx.controls.TextArea;

my_ta.setSize(320, 240);
my_ta.wordWrap = true;
my_ta.vScrollPolicy = "off";

// Chargement du texte à afficher et définition du gestionnaire onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_ta.text = src;
    } else {
        my_ta.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

TextArea.wordWrap

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textAreaInstance.wordWrap

Description

Propriété : valeur booléenne qui indique si le texte est renvoyé à la ligne automatiquement (`true`) ou non (`false`). La valeur par défaut est `true`.

REMARQUE	Si vous créez une occurrence TextArea à l'aide de la méthode `createClassObject()`, la valeur par défaut pour wordWrap est `false`.

Exemple

L'exemple suivant définit la propriété `wordwrap` sur `false` pour le composant TextArea appelé `my_ta` de façon à ce qu'une barre de défilement horizontale accède au texte au-delà des limites latérales.

Vous devez d'abord ajouter une occurrence du composant TextArea sur la scène et la nommer **my_ta** ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextArea sur la scène (nom d'occurrence : my_ta)
*/

var my_ta:mx.controls.TextArea;

my_ta.setSize(320, 240);
my_ta.wordWrap = false;
// my_ta.vScrollPolicy = "off";

// Chargement du texte à afficher et définition du gestionnaire onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_ta.text = src;
    } else {
        my_ta.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

CHAPITRE 46

Composant TextInput

46

TextInput est un composant à une seule ligne qui renvoie à la ligne automatiquement l'objet TextField ActionScript natif. Vous pouvez utiliser les styles pour personnaliser le composant TextInput. Lorsqu'une occurrence est désactivée, son contenu s'affiche dans une couleur représentée par le style disabledColor. Un composant TextInput peut également être formaté en HTML ou en tant que champ de mot de passe masquant le texte.

REMARQUE

Un composant TextInput est pris en charge pour ActionScript 2.0 et ActionScript 3.0. Ce document présente le composant de la version 2. Si vous utilisez celui de la version 3, reportez-vous à « Utilisation du composant CheckBox » dans *Utilisation des composants ActionScript 3.0*.

Un composant TextInput peut être activé ou désactivé dans une application. Lorsqu'il est désactivé, il ne reçoit pas les informations en provenance de la souris ou du clavier. Lorsqu'il est activé, il suit les mêmes règles de focus, de sélection et de navigation qu'un objet TextField ActionScript. Lorsqu'une occurrence de TextInput a le focus, vous pouvez également utiliser les touches suivantes pour le contrôler :

Touche	Description
Touches fléchées	Déplacent le point d'insertion d'un caractère vers la gauche et vers la droite.
Maj+Tab	Place le focus sur l'objet précédent.
Tab	Place le focus sur l'objet suivant.

Pour plus d'informations sur le contrôle du focus, reportez-vous à « Classe FocusManager », à la page 745 ou à « Création de la navigation personnalisée du focus » dans *Utilisation des composants ActionScript 2.0*.

L'aperçu en direct des occurrences de TextInput reflète les modifications apportées aux paramètres dans l'inspecteur Propriétés ou l'inspecteur des composants pendant la programmation. Lors d'un aperçu en direct, il n'est pas possible de sélectionner du texte et vous ne pouvez pas entrer de texte dans l'occurrence du composant sur la scène.

Lorsque vous ajoutez un composant TextInput à une application, vous pouvez utiliser le panneau Accessibilité pour le rendre accessible aux lecteurs d'écran.

Utilisation du composant TextInput

Vous pouvez utiliser un composant TextInput là où vous avez besoin d'un champ de texte à une seule ligne. Si vous avez besoin d'un champ de texte multiligne, utilisez le Composant TextArea. Par exemple, vous pouvez utiliser un composant TextInput en tant que champ de mot de passe dans un formulaire. Vous pouvez également définir un écouteur qui vérifie si le champ comporte suffisamment de caractères lorsque l'utilisateur sort du champ. Cet écouteur peut afficher un message d'erreur indiquant que l'utilisateur n'a pas entré le nombre de caractères adéquat.

Paramètres de TextInput

Vous pouvez définir les paramètres de création suivants pour chaque occurrence du composant Menu dans
l'inspecteur des propriétés ou l'inspecteur des composants (Fenêtre > Inspecteur de composants) :

editable indique si le composant TextInput est modifiable (`true`) ou non (`false`). La valeur par défaut est `true`.

password indique si le champ est un champ de mot de passe (`true`) ou non (`false`). La valeur par défaut est `false`.

text spécifie le contenu du composant TextInput. Vous ne pouvez pas entrer de retour chariot dans l'inspecteur Propriétés ou l'inspecteur des composants. La valeur par défaut est "" (une chaîne vide).

Dans l'inspecteur des composants, vous pouvez définir les paramètres supplémentaires suivants pour chaque occurrence du composant TextInput (Fenêtre > Inspecteur de composants) :

maxChars est le nombre maximum de caractères que le champ de texte peut contenir. La valeur par défaut est `null` (c'est-à-dire illimitée).

restrict indique le jeu de caractères qu'un utilisateur peut entrer dans le champ de texte. La valeur par défaut est `undefined`. Voir « TextInput.restrict », à la page 1278.

enabled est une valeur booléenne qui indique si le composant peut recevoir le focus et l'entrée. La valeur par défaut est `true`.

visible est une valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`). La valeur par défaut est `true`.

REMARQUE	Les propriétés minHeight et minWidth sont utilisées par des routines de dimensionnement internes. Elles sont définies dans UIObject et remplacées par différents composants selon vos besoins. Ces propriétés peuvent être utilisées pour créer un gestionnaire de dispositions personnalisées. Dans le cas contraire, la définition de ces propriétés dans l'inspecteur de composants n'a aucun effet visible.

Vous pouvez contrôler ces options et d'autres options du composant TextInput à l'aide des propriétés, méthodes et événements d'ActionScript. Pour plus d'informations, voir « Classe TextInput », à la page 1263.

Création d'une application avec le composant TextInput

La procédure suivante explique comment ajouter un composant TextInput à une application pendant la programmation. Dans cet exemple, le composant est un champ de mot de passe avec un écouteur d'événements qui détermine si l'utilisateur a entré le nombre de caractères adéquat.

Pour créer une application avec le composant TextInput :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser un composant TextInput du panneau Composants jusqu'à la scène.
3. Dans l'inspecteur Propriétés, procédez comme suit :
   - Entrez le nom d'occurrence **my_ti**.
   - Laissez le paramètre text vide.
   - Définissez le paramètre editable `true`.
4. Sélectionnez l'image 1 dans le scénario, ouvrez le panneau Actions et saisissez le code suivant :

```
/**
 Requiert :
  - Occurrence TextInput sur la scène (nom d'occurrence : my_ti)
*/

var my_ti:mx.controls.TextInput;

// Création d'un objet écouteur.
var tiListener:Object = new Object();
tiListener.handleEvent = function (evt_obj:Object){
 if (evt_obj.type == "enter"){
```

```
    if (my_ti.length < 8) {
     trace("You must enter at least 8 characters");
    } else {
     trace("Thanks");
    }
   }
}
// Ajout de l'écouteur.
my_ti.addEventListener("enter", tiListener);
```

Ce code configure un gestionnaire d'événements `enter` sur l'occurrence de composant TextInput appelée `my_ti`. Si l'utilisateur tape moins de huit caractères, l'exemple affiche le message suivant : `Vous devez entrer au moins 8 caractères`. Si l'utilisateur huit caractères ou plus, l'exemple affiche : `Merci`.

5. Une fois que le texte est entré dans l'occurrence `my_ti`, vous pouvez obtenir sa valeur comme suit :

```
var my_text:String = my_ti.text;
```

Personnalisation du composant TextInput

Vous pouvez transformer un composant TextInput horizontalement pendant la programmation et lors de l'exécution. Lors de la programmation, sélectionnez le composant sur la scène et utilisez l'outil Transformer librement ou une commande Modification > Transformer. Lors de l'exécution, utilisez `UIObject.setSize()` ou toute propriété ou méthode applicable de la Classe TextInput.

Lorsqu'un composant TextInput est redimensionné, la bordure prend la taille du nouveau cadre de sélection. Le composant TextInput n'utilise pas de barres de défilement, mais le point d'insertion défile automatiquement lorsque l'utilisateur intervient sur le texte. Le champ de texte est alors redimensionné dans la zone restante. Les éléments d'un composant TextInput n'ont pas de taille fixe. Si le composant TextInput est trop petit pour afficher le texte, le texte est rogné.

Utilisation de styles avec le composant TextInput

Les propriétés de style `backgroundColor` et `borderStyle` du composant TextInput sont définies sur une déclaration de style de classe. Les styles de classe remplacent les styles globaux. Pour définir les propriétés de style `backgroundColor` et `borderStyle`, vous devez donc créer une autre déclaration de style personnalisée ou la définir sur l'occurrence.

Un composant TextInput prend en charge les styles suivants :

Style	Thème	Description
`backgroundColor`	Les deux	Couleur d'arrière-plan. La couleur par défaut est le blanc.
`borderStyle`	Les deux	Le composant TextInput utilise une occurrence de RectBorder en tant que bordure et répond aux styles définis sur cette classe. Voir « Classe RectBorder », à la page 1111. Le style de bordure par défaut est `"inset"`.
`marginLeft`	Les deux	Nombre indiquant la marge gauche pour le texte. La valeur par défaut est 0.
`marginRight`	Les deux	Nombre indiquant la marge droite pour le texte. La valeur par défaut est 0.
`color`	Les deux	Couleur du texte. La valeur par défaut est 0x0B333C pour le thème Halo et vide pour le thème Sample.
`disabledColor`	Les deux	Couleur du texte lorsque le composant est désactivé. La couleur par défaut est 0x848384 (gris foncé).
`embedFonts`	Les deux	Valeur booléenne qui indique si la police spécifiée dans `fontFamily` est de type intégré. Ce style doit être défini sur `true` si `fontFamily` fait référence à une police intégrée. Dans le cas contraire, la police intégrée n'est pas utilisée. Si ce style est défini sur `true` et que `fontFamily` ne fait pas référence à une police intégrée, aucun texte n'est affiché. La valeur par défaut est `false`.
`fontFamily`	Les deux	Nom de la police du texte. La valeur par défaut est `"_sans"`.
`fontSize`	Les deux	Taille, en points, de la police. La valeur par défaut est`10`.
`fontStyle`	Les deux	Style de police : `"normal"` ou `"italic"`. La valeur par défaut est `"normal"`.

Style	Thème	Description
`fontWeight`	Les deux	Epaisseur de la police : `"none"` ou `"bold"`. La valeur par défaut est `"none"`. Tous les composants peuvent également accepter la valeur `"normal"` au lieu de `"none"` pendant un appel à la méthode `setStyle()`, mais les appels suivants à `getStyle()` renvoient `"none"`.
`textAlign`	Les deux	Alignement du texte : `"left"`, `"right"` ou `"center"`. La valeur par défaut est `"left"`.
`textIndent`	Les deux	Nombre indiquant le retrait du texte. La valeur par défaut est 0.
`textDecoration`	Les deux	Décoration du texte : `"none"` ou `"underline"`. La valeur par défaut est `"none"`.

Les composants TextArea et TextInput utilisent les mêmes styles et sont souvent utilisés de la même façon. Par conséquent, ils partagent par défaut la même déclaration de style de niveau de classe. Par exemple, le code suivant définit un style sur la déclaration TextArea mais il affecte à la fois les composants TextInput et TextArea.

```
_global.styles.TextArea.setStyle("disabledColor", 0xBBBBFF);
```

Pour séparer les composants et fournir des styles de niveau de classe pour l'un et non pour l'autre, créez une déclaration de style.

```
import mx.styles.CSSStyleDeclaration;
_global.styles.TextInput = new CSSStyleDeclaration();
_global.styles.TextInput.setStyle("disabledColor", 0xFFBBBB);
```

Notez que cet exemple ne vérifie pas si `_global.styles.TextInput` existait avant d'être remplacé ; dans cet exemple, vous savez qu'il existe et vous souhaitez le remplacer.

Utilisation d'enveloppes avec le composant TextInput

Le composant TextArea utilise une occurrence de RectBorder pour sa bordure. Pour plus d'informations sur l'application d'enveloppe à ces éléments visuels, reportez-vous à « Classe RectBorder », à la page 1111.

Classe TextInput

Héritage MovieClip > Classe UIObject > Classe UIComponent > TextInput

Nom de classe ActionScript mx.controls.TextInput

Les propriétés de la classe TextInput vous permettent de définir le contenu, le formatage et la position horizontale du texte lors de l'exécution. Vous pouvez également indiquer si le champ est modifiable et s'il s'agit d'un champ Mot de passe. Vous pouvez également limiter le nombre de caractères que l'utilisateur peut saisir.

La définition d'une propriété de la classe TextInput avec ActionScript annule le paramètre du même nom défini dans l'inspecteur Propriétés ou des composants.

Le composant TextInput utilise le gestionnaire de focus pour remplacer le rectangle de focus par défaut de Flash Player et tracer un rectangle de focus personnalisé aux coins arrondis. Pour plus d'informations, voir « Classe FocusManager », à la page 745.

Le composant TextInput prend en charge les styles CSS et tout style HTML pris en charge par Flash Player. Pour plus d'informations sur la prise en charge de CSS, consultez les spécifications du W3C sur le site www.w3.org/TR/REC-CSS2/.

Vous pouvez manipuler la chaîne de texte en utilisant la chaîne renvoyée par l'objet texte.

Toutes les classes de composants ont une propriété `version` qui correspond à une propriété de classe. Les propriétés de classe ne sont disponibles que pour la classe elle-même. La propriété `version` renvoie une chaîne qui indique la version du composant. Pour accéder à cette propriété, utilisez le code suivant :

```
trace(mx.controls.TextInput.version);
```

REMARQUE

Le code `trace(myTextInputInstance.version);` renvoie `undefined`.

Méthodes de la classe TextInput

La classe TextInput ne présente pas de méthodes exclusives.

Méthodes héritées de la classe UIObject

Le tableau suivant répertorie les méthodes de la classe TextInput héritées de la classe UIObject. Lorsque vous appelez ces méthodes depuis l'objet TextInput, utilisez le formulaire *`TextInputInstance.methodName`*.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image active.
`UIObject.setSize()`	Redimensionne l'objet à la taille demandée.
`UIObject.setSkin()`	Définit une enveloppe dans l'objet.
`UIObject.setStyle()`	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant répertorie les méthodes de la classe TextInput héritées de la classe UIComponent. Lorsque vous appelez ces méthodes depuis l'objet TextInput, utilisez le formulaire *`TextInputInstance.methodName`*.

Méthode	Description
`UIComponent.getFocus()`	Renvoie une référence à l'objet ayant le focus.
`UIComponent.setFocus()`	Attribue le focus à l'occurrence de composant.

Propriétés de la classe TextInput

Le tableau suivant répertorie les propriétés de la classe TextInput.

Propriété	Description
`TextInput.editable`	Valeur booléenne indiquant si le champ est modifiable (`true`) ou non (`false`).
`TextInput.hPosition`	Position de défilement du texte horizontalement dans le champ.
`TextInput.length`	Lecture seule ; le nombre de caractères dans un composant TextInput.
`TextInput.maxChars`	Nombre maximum de caractères que l'utilisateur peut entrer dans un champ de texte.
`TextInput.maxHPosition`	Lecture seule ; la valeur maximum possible pour `TextField.hPosition`.
`TextInput.password`	Valeur booléenne qui indique si le champ de texte est un champ de mot de passe qui masque les caractères saisis ou non.
`TextInput.restrict`	Indique les caractères que l'utilisateur peut entrer dans un champ de texte.
`TextInput.text`	Définit le contenu du texte d'un composant TextInput.

Propriétés héritées de la classe UIObject

Le tableau suivant répertorie les propriétés de la classe TextInput héritées de la classe UIObject. Lorsque vous accédez à ces propriétés depuis l'objet TextInput, utilisez le formulaire *TextInputInstance.propertyName*.

Propriété	Description
UIObject.bottom	Lecture seule ; position du bord inférieur de l'objet par rapport au bord inférieur de son parent.
UIObject.height	Lecture seule ; hauteur de l'objet, en pixels.
UIObject.left	Lecture seule ; bord gauche de l'objet, en pixels.
UIObject.right	Lecture seule ; position du bord droit de l'objet par rapport au bord droit de son parent.
UIObject.scaleX	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
UIObject.scaleY	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
UIObject.top	Lecture seule ; renvoie la position du bord supérieur de l'objet par rapport à son parent.
UIObject.visible	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
UIObject.width	Lecture seule ; largeur de l'objet, en pixels.
UIObject.x	Lecture seule ; bord gauche de l'objet, en pixels.
UIObject.y	Lecture seule ; bord supérieur de l'objet, en pixels.

Propriétés héritées de la classe UIComponent

Le tableau suivant répertorie les propriétés de la classe TextInput héritées de la classe UIComponent. Lorsque vous accédez à ces propriétés depuis l'objet TextInput, utilisez le formulaire *TextInputInstance.propertyName*.

Propriété	Description
UIComponent.enabled	Indique si le composant peut recevoir le focus et la saisie.
UIComponent.tabIndex	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Evénements de la classe TextInput

Le tableau suivant répertorie les événements de la classe TextInput.

Evénement	Description
`TextInput.change`	Diffusé lorsque le champ TextInput est modifié.
`TextInput.enter`	Diffusé lorsque la touche Entrée est enfoncée.

Evénements hérités de la classe UIObject

Le tableau suivant répertorie les événements de la classe TextInput hérités de la classe UIObject.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant répertorie les événements de la classe TextInput hérités de la classe UIComponent.

Evénement	Description
`UIComponent.focusIn`	Diffusé lorsqu'un objet reçoit le focus.
`UIComponent.focusOut`	Diffusé lorsqu'un objet perd le focus.
`UIComponent.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIComponent.keyUp`	Diffusé lorsqu'une touche est relâchée.

TextInput.change

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.change = function(eventObject:Object) {
  //...
};
textInputInstance.addEventListener("change", listenerObject)
```

Utilisation 2 :

```
on (change){
  //...
}
```

Description

Evénement : indique aux écouteurs que le texte a été modifié. Cet événement est diffusé après la modification du texte. Cet événement ne peut être utilisé pour empêcher l'ajout de certains caractères au champ de texte du composant. Utilisez plutôt `TextInput.restrict`. L'événement est déclenché uniquement par l'action de l'utilisateur et non par une modification par programme.

Le premier exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence TextInput. Le mot-clé `this`, utilisé dans un gestionnaire `on()` relié à un composant, fait référence à l'occurrence de ce composant. Par exemple, le code suivant, associé à l'occurrence `myTextInput`, envoie « _level0.myTextInput » vers le panneau Sortie :

```
on (change){
  trace(this);
}
```

Le second exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*textInputInstance*) distribue un événement (dans ce cas, change) qui est géré par une fonction (également appelée *gestionnaire*) sur un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode EventDispatcher.addEventListener() sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

Cet exemple crée un écouteur pour un événement change sur l'occurrence de composant TextInput my_ti. Lorsqu'un événement change a lieu, l'exemple affiche « Input has changed ».

Vous devez d'abord faire glisser une occurrence du composant TextInput sur la scène et la nommer my_ti ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextInput sur la scène (nom d'occurrence : my_ti)
*/

var my_ti:mx.controls.TextInput;

// Création d'un objet écouteur.
var tiListener:Object = new Object();
tiListener.change = function(evt_obj:Object) {
 trace("Input has changed");
};
// Ajout de l'écouteur.
my_ti.addEventListener("change", tiListener);
```

Voir aussi

EventDispatcher.addEventListener()

TextInput.editable

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textInputInstance.editable

Description

Propriété : valeur booléenne qui indique si le composant est modifiable (true) ou non (false). La valeur par défaut est true.

Exemple

Cet exemple définit la propriété editable sur une valeur false prou l'occurrence de composant TextInput my_ti. Ceci empêche l'utilisateur de saisir du texte dans l'occurrence. Vous pouvez définir la propriété sur true pour pouvoir modifier l'occurrence TextInput.

Vous devez d'abord faire glisser une occurrence du composant TextInput sur la scène et la nommer my_ti ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextInput sur la scène (nom d'occurrence : my_ti)
*/

var my_ti:mx.controls.TextInput;

my_ti.editable = false;
```

TextInput.enter

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.enter = function(eventObject:Object) {
  //...
};
textInputInstance.addEventListener("enter", listenerObject);
```

Utilisation 2 :

```
on (enter) {
  //...
}
```

Description

Evénement : indique aux écouteurs que l'utilisateur a appuyé sur la touche Entrée.

Le premier exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence TextInput. Le mot-clé `this`, utilisé dans un gestionnaire `on()` relié à un composant, fait référence à l'occurrence de ce composant. Par exemple, le code suivant, associé à l'occurrence `myTextInput`, envoie « _level0.myTextInput » vers le panneau Sortie :

```
on (enter){
  trace(this);
}
```

Le second exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`textInputInstance`*) distribue un événement (dans ce cas, `enter`) qui est géré par une fonction (également appelée *gestionnaire*) sur un objet écouteur (*`listenerObject`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`eventObject`*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

Cet exemple crée un écouteur pour un événement enter sur une occurrence de composant TextInput appelée `my_ti`. Lorsque l'événement enter a lieu, si l'utilisateur a saisi moins de huit caractères, l'exemple affiche : `Vous devez entrer au moins 8 caractères`. Sinon, il affiche `Merci !`

Vous devez d'abord faire glisser une occurrence du composant TextInput sur la scène et la nommer `my_ti` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextInput sur la scène (nom d'occurrence : my_ti)
*/

var my_ti:mx.controls.TextInput;

// Création d'un objet écouteur.
var tiListener:Object = new Object();
tiListener.handleEvent = function (evt_obj:Object){
 if (evt_obj.type == "enter"){
  if (my_ti.length < 8) {
   trace("You must enter at least 8 characters");
  } else {
   trace("Thanks");
  }
 }
}
// Ajout de l'écouteur.
my_ti.addEventListener("enter", tiListener);
```

Voir aussi

`EventDispatcher.addEventListener()`

TextInput.hPosition

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textInputInstance.hPosition

Description

Propriété ; indique le nombre de pixels que vous avez fait défiler pour que le champ de texte du composant TextInput contienne l'entrée de l'utilisateur. La valeur par défaut est 0.

REMARQUE	La valeur change pour le même texte sur différents ordinateurs en raison des caractéristiques du moniteur, de la taille de l'écran et de la police.

Exemple

L'exemple suivant crée un écouteur pour un événement `change` sur l'occurrence de composant TextInput appelée `my_ti`. L'écouteur accède à la propriété `hPosition` pour afficher la position actuelle pour chaque caractère entré par l'utilisateur.

Vous devez d'abord faire glisser une occurrence du composant TextInput sur la scène et la nommer `my_ti` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextInput sur la scène (nom d'occurrence : my_ti)
*/

var my_ti:mx.controls.TextInput;

// Création d'un objet écouteur.
var tiListener:Object = new Object();
tiListener.change = function(evt_obj:Object) {
 trace("hPosition = " + my_ti.hPosition);
};
// Ajout de l'écouteur.
my_ti.addEventListener("change", tiListener);
```

TextInput.length

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textInputInstance.length

Description

Propriété en lecture seule ; un nombre qui indique le nombre de caractères dans un composant TextInput. Un caractère tel que tab ("\t") compte comme un seul caractère. La valeur par défaut est 0.

Exemple

L'exemple suivant crée un écouteur pour un événement change sur l'occurrence de composant TextInput appelée my_ti. L'écouteur accède à la propriété length pour afficher la longueur du texte dans my_ti lorsque l'utilisateur saisit du texte.

Vous devez d'abord faire glisser une occurrence du composant TextInput sur la scène et la nommer my_ti ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextInput sur la scène (nom d'occurrence : my_ti)
*/

var my_ti:mx.controls.TextInput;

// Création d'un objet écouteur.
var tiListener:Object = new Object();
tiListener.change = function(evt_obj:Object) {
 trace("Length of text: " + my_ti.length);
};
// Ajout de l'écouteur.
my_ti.addEventListener("change", tiListener);
```

TextInput.maxChars

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textInputInstance.maxChars

Description

Propriété : nombre maximum de caractères qu'un champ de texte peut contenir. Un script peut insérer plus de texte que la propriété `maxChars` ne le permet ; cette propriété n'indique que la quantité de texte qu'un utilisateur peut entrer. Si la valeur de cette propriété est `null`, il n'y a pas de limite sur la quantité de texte qu'un utilisateur peut entrer. La valeur par défaut est `null`.

Exemple

L'exemple suivant limite à huit le nombre de caractères qu'un utilisateur peut saisir dans l'occurrence de composant TextInput appelée `my_ti`. Il définit également la propriété `password` qui masque les caractères entrés en affichant un astérisque à la place du caractère saisi.

Vous devez d'abord faire glisser une occurrence du composant TextInput sur la scène et la nommer `my_ti` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextInput sur la scène (nom d'occurrence : my_ti)
*/

var my_ti:mx.controls.TextInput;

my_ti.maxChars = 8;
my_ti.password = true;
```

TextInput.maxHPosition

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textInputInstance.maxHPosition

Description

Propriété en lecture seule : la valeur de `maxHPosition` est la position (en pixels) du caractère visible lorsque le curseur est déplacé tout à fait à droite du texte. Il ne s'agit pas de la position (en pixels) du dernier caractère. Il s'agit de la position (en pixels) jusqu'à la droite du dernier caractère dans le champ TextInput. La valeur par défaut est 0.

Vous devez d'abord faire glisser une occurrence du composant TextInput sur la scène et la nommer `my_ti` ; ajoutez ensuite le code suivant à l'image 1.

Exemple

Le code suivant crée un écouteur pour un événement change sur l'occurrence de composant TextInput appelée `my_ti`. Lorsque l'événement change a lieu, l'écouteur affiche les valeurs `hPosition` et `maxHPosition` actuelles pour chaque caractère entré par l'utilisateur :

```
/**
 Requiert :
  - Occurrence TextInput sur la scène (nom d'occurrence : my_ti)
*/

var my_ti:mx.controls.TextInput;

// Création d'un objet écouteur.
var tiListener:Object = new Object();
tiListener.change = function(evt_obj:Object) {
 trace("hPosition: " + my_ti.hPosition + " of " + my_ti.maxHPosition);
};
// Ajout de l'écouteur.
my_ti.addEventListener("change", tiListener);
```

TextInput.password

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textInputInstance.password

Description

Propriété : valeur booléenne indiquant si le champ de texte est un champ de mot de passe (`true`) ou non (`false`). Si la valeur de cette propriété est `true`, le champ de texte est un champ de texte de mot de passe dont le contenu est masqué. Si la valeur de cette propriété est `false`, le champ de texte n'est pas un champ de mode passe. La valeur par défaut est `false`.

Exemple

L'exemple suivant définit la propriété `password` pour qu'elle affiche un astérisque à la place du caractère entré par l'utilisateur dans l'occurrence de composant TextInput appelée `my_ti`. Il définit également `maxChars` pour limiter à huit le nombre maximum de caractères que l'utilisateur peut entrer.

Vous devez d'abord faire glisser une occurrence du composant TextInput sur la scène et la nommer `my_ti` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextInput sur la scène (nom d'occurrence : my_ti)
*/

var my_ti:mx.controls.TextInput;

my_ti.maxChars = 8;
my_ti.password = true;
```

TextInput.restrict

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textInputInstance.restrict

Description

Propriété : indique le jeu de caractères qu'un utilisateur peut entrer dans le champ de texte. La valeur par défaut est `undefined`. Si la valeur de cette propriété est `null` ou une chaîne vide (`""`), un utilisateur peut entrer n'importe quel caractère. Si cette propriété est une chaîne de caractères, l'utilisateur peut entrer uniquement les caractères de la chaîne ; la chaîne est lue de gauche à droite. Vous pouvez spécifier une plage en utilisant un tiret (-).

Si la chaîne commence par ^, tous les caractères qui suivent le ^ sont considérés comme inacceptables. Si la chaîne ne commence pas par ^, les caractères dans la chaîne sont considérés comme acceptables. Le ^ peut également permettre de basculer entre des caractères acceptables et inacceptables.

Par exemple, le code suivant permet d'entrer les caractères A à Z sauf X et Q :

```
Ta.restrict = "A-Z^XQ";
```

Vous pouvez utiliser la barre oblique inverse (\) pour entrer un trait d'union (-), un caret (^) ou une barre oblique inverse (\), comme indiqué ici :

```
\^
\-
\\
```

Le fait de placer une barre oblique inverse (\) entre guillemets dans le panneau Actions a une signification particulière pour l'interpréteur de guillemets du panneau. Cette syntaxe indique en effet que le caractère suivant la barre oblique inverse (\) doit être traité comme tel. Par exemple, vous pouvez utiliser le code suivant pour entrer un guillemet droit simple :

```
var leftQuote = "\'";
```

L'interpréteur restrict du panneau Actions utilise également la barre oblique inverse (\) comme caractère d'échappement. Vous penserez donc peut-être que la chaîne suivante est correcte :

```
myText.restrict = "0-9\-\^\\";
```

Toutefois, comme l'expression est placée entre guillemets, la valeur suivante est envoyée à l'interpréteur restrict : `0-9-^\`, et cette valeur ne peut pas être interprétée.

L'interpréteur restrict exigeant l'utilisation de guillemets, vous devez impérativement utiliser le caractère d'échappement pour que l'expression soit traitée correctement par l'interpréteur de guillemets intégré du panneau Actions. Pour envoyer la valeur `0-9\-\^\\` à l'interpréteur restrict, vous devez donc entrer le code suivant :

```
myText.restrict = "0-9\\-\\^\\\\";
```

La propriété `restrict` limite uniquement l'interaction de l'utilisateur, un script pouvant insérer n'importe quel texte dans le champ de texte. Cette propriété ne se synchronise pas avec les cases à cocher de polices vectorielles intégrées de l'inspecteur Propriétés.

Exemple

L'exemple suivant fournit trois différentes utilisations de la propriété `restrict`. Le premier exemple d'utilisation limite la saisie aux caractères en majuscules de A à Z, aux espaces et aux chiffres. Le deuxième exemple d'utilisation autorise tous les caractères excepté les caractères en minuscules de a à z. Le troisième autorise uniquement les chiffres, -, ^ et \.

Vous devez d'abord faire glisser une occurrence du composant TextInput sur la scène et la nommer `my_ti` ; ajoutez ensuite le code à l'image 1, en utilisant une seule des instructions `restrict` suivantes à la fois.

```
/**
 Requiert :
  - Occurrence TextInput sur la scène (nom d'occurrence : my_ti)
*/

var my_ti:mx.controls.TextInput;

// Exemple 1 : Autorise uniquement les majuscules A-Z, les espaces et
// les chiffres de 0 à 9.
my_ti.restrict = "A-Z 0-9";

// Exemple 2 : Autorise tous les caractères SAUF les minuscules de a à z.
my_ti.restrict = "^a-z";

// Exemple 3 : Autorise uniquement les chiffres de 0-9, les tirets (-),
// les signes ^ et \
my_ti.restrict = "0-9\\-\\^\\\\";
```

TextInput.text

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

textInputInstance.text

Description

Propriété : le contenu du texte d'un composant TextInput. La valeur par défaut est "" (une chaîne vide).

Exemple

Le code suivant permet de placer une chaîne dans l'occurrence TextInput, my_ti, et de présenter cette chaîne dans le panneau Sortie.

Vous devez d'abord faire glisser une occurrence du composant TextInput sur la scène et la nommer my_ti ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Occurrence TextInput sur la scène (nom d'occurrence : my_ti)
*/

var my_ti:mx.controls.TextInput;

my_ti.text = "The Royal Nonesuch";
trace(my_ti.text); // « The Royal Nonesuch ».
```

CHAPITRE 47

47 Interface TransferObject

Nom de classe ActionScript mx.data.to.TransferObject

L'interface TransferObject définit un ensemble de méthodes que les éléments gérés par le composant DataSet doivent implémenter. La propriété `DataSet.itemClassName` précise le nom de la classe d'objets de transfert qui est instanciée chaque fois qu'un nouvel élément est nécessaire. Vous pouvez également spécifier cette propriété pour un composant DataSet sélectionné à l'aide de l'inspecteur Propriétés.

REMARQUE	L'interface TransferObject est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Méthodes de l'interface TransferObject

Le tableau suivant répertorie les méthodes de l'interface TransferObject.

Méthode	Description
`TransferObject.clone()`	Crée une occurrence de l'objet de transfert.
`TransferObject.getPropertyData()`	Renvoie les données de cet objet de transfert.
`TransferObject.setPropertyData()`	Définit les données de cet objet de transfert.

TransferObject.clone()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
class itemClass implements mx.data.to.TransferObject {
  function clone() {
    // Votre code ici.
  }
}
```

Paramètres

Aucun.

Renvoie

Une copie de l'objet de transfert.

Description

Méthode : crée une occurrence de l'objet de transfert. La mise en place de cette méthode permet de créer une copie de l'objet de transfert existant et de ses propriétés, puis de renvoyer cet objet.

Exemple

La fonction suivante renvoie une copie de cet objet de transfert et de toutes les propriétés définies sur les mêmes valeurs que l'original :

```
class itemClass implements mx.data.to.TransferObject {
  function clone():Object {
    var copy:itemClass = new itemClass();
    for (var p in this) {
      copy[p]= this[p];
    }
    return(copy);
  }
}
```

TransferObject.getPropertyData()

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

```
class itemClass implements mx.data.to.TransferObject {
  function getPropertyData() {
  // Votre code ici.
  }
}
```

Paramètres

Aucun.

Renvoie

Un objet.

Description

Méthode : renvoie les données de cet objet de transfert. La mise en place de cette méthode peut permettre de renvoyer un objet ActionScript anonyme avec les propriétés et valeurs correspondantes.

Exemple

La fonction suivante renvoie un objet nommé `internalData` qui contient les propriétés et les valeurs de l'objet Contact :

```
class Contact implements mx.data.to.TransferObject {
function getPropertyData():Object {
  var internalData:Object = {name:name, readOnly:_readOnly, phone:phone,
  zip:zip.zipPlus4};
  return(internalData);
}
```

TransferObject.setPropertyData()

Disponibilité

Flash Player 7.

Edition

Flash MX 2004.

Utilisation

```
class yourClass implements TransferObject {
  function setPropertyData(propData) {
  // Votre code ici.
  }
}
```

Paramètres

propData Objet contenant les données affectées à cet objet de transfert.

Renvoie

Aucune.

Description

Méthode : définit les données de cet objet de transfert. Le paramètre *propData* est un objet dont les champs contiennent les données affectées par le composant DataSet à cet objet de transfert.

Exemple

La fonction suivante reçoit un paramètre *propData* et applique les valeurs de ses propriétés à celles de l'objet Contact :

```
class Contact implements mx.data.to.TransferObject {

  function setPropertyData(propData: Object):Void {
    _readOnly = propData.readOnly;
    phone = propData.phone;
    zip = new mx.data.types.ZipCode(data.zip);
  }

  public var name:String;
  public var phone:String;
  public var zip:ZipCode;
  private var _readOnly:Boolean; // indication du caractère inaltérable
                                 // ou non
}
```

CHAPITRE 48

Classe TransitionManager

48

Nom de classe ActionScript mx.transitions.TransitionManager

La classe TransitionManager et les classes basées sur la classe Transition qui définissent les effets vous permettent d'appliquer rapidement des effets de transition impressionnants aux diapositives et aux clips.

REMARQUE

La classe TransitionManager est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Comme son nom l'indique, la classe TransitionManager gère des transitions. Elle vous permet d'appliquer l'un des dix effets d'animation à des diapositives et à des clips. Pour créer des composants personnalisés, vous pouvez utiliser la classe TransitionManager et appliquer des effets d'animation à des clips dans l'interface visuelle de votre composant. Les effets de transition sont définis dans un ensemble de classes de transition qui étendent toutes la classe de base `mx.transitions.Transition`. Vous devez appliquer les transitions uniquement dans une occurrence TransitionManager ; ne les instanciez pas directement. La classe TransitionManager implémente des événements animation.

Utilisation de la classe TransitionManager

Pour utiliser les méthodes et les propriétés de la classe TransitionManager, vous disposez de deux possibilités pour créer une occurrence. La plus simple consiste à appeler la méthode `TransitionManager.start()`, qui crée une occurrence TransitionManager, désigne l'objet cible, applique une transition avec une méthode d'accélération et la démarre en un appel. Le code suivant utilise la méthode `TransitionManager.start()`.

```
mx.transitions.TransitionManager.start(myMovieClip_mc,
  {type:mx.transitions.Zoom, direction:mx.transitions.Transition.IN,
  duration:1, easing:mx.transitions.easing.Bounce.easeOut});
```

Pour plus d'informations sur la méthode `TransitionManager.start()`, son utilisation et ses paramètres, reportez-vous à « TransitionManager.start() » à la page 1294.

Vous avez également la possibilité de créer une occurrence de la classe TransitionManager à l'aide de l'opérateur new. Vous pouvez ensuite désigner les propriétés de transition, puis démarrer l'effet de transition en appelant la méthode `TransitionManager.startTransition()`. Le code suivant utilise la méthode `TransitionManager.startTransition()`.

```
var myTransitionManager:mx.transitions.TransitionManager = new
  mx.transitions.TransitionManager(myMovieClip_mc);
myTransitionManager.startTransition({type:mx.transitions.Zoom,
  direction:Transition.IN, duration:1,
  easing:mx.transitions.easing.Bounce.easeOut});
```

Paramètres de la classe TransitionManager

Pour créer une occurrence d'une classe TransitionManager à l'aide de l'opérateur *new*, vous devez désigner un clip cible dans le paramètre `content` correspondant à son constructeur. Le constructeur pour la classe mx.transitions.TransitionManager possède le nom et le type de paramètre suivants :

```
TransitionManager(content:MovieClip)
```

content représente l'objet clip auquel l'occurrence TransitionManager applique une transition.

REMARQUE

Si vous créez une occurrence TransitionManager en utilisant l'opérateur new, vous devez ensuite désigner les propriétés de la transition que vous souhaitez appliquer, puis appelez la méthode `TransitionManager.startTransition()` pour lancer la transition, faute de quoi elle ne sera pas appliquée à un clip ni lancée. Pour plus d'informations sur la méthode `TransitionManager.startTransition()`, son utilisation et ses paramètres, reportez-vous à « TransitionManager.startTransition() » à la page 1295. Au lieu d'exécuter cette procédure à deux étapes, vous pouvez créer rapidement une occurrence TransitionManager en appelant simplement la méthode `TransitionManager.start()`. Pour plus d'informations, reportez-vous à « TransitionManager.start() » à la page 1294. Cette méthode vous permet de créer une occurrence TransitionManager, de fournir le clip cible et de spécifier les propriétés de transition en une seule étape.

Définition d'une classe d'accélération et d'une méthode dans une transition

Si vous créez une occurrence de la classe TransitionManager en utilisant la méthode `TransitionManager.start()`, vous utilisez la propriété d'accélération du paramètre `transParam` pour spécifier une fonction ou une méthode qui fournit un calcul d'accélération. Pour consulter la description complète des classes et des méthodes d'accélération disponibles, reportez-vous à « Présentation des classes et des méthodes d'accélération », à la page 1366

Récapitulatif de la classe TransitionManager

Les sections ci-dessous répertorient les méthodes, les propriétés et les événements de la classe TransitionManager.

Récapitulatif des méthodes de la classe TransitionManager

Le tableau suivant répertorie les méthodes de la classe TransitionManager.

Méthode	Description
`TransitionManager.start()`	Crée une occurrence de la classe TransitionManager, désigne l'objet cible, applique une transition et la lance.
`TransitionManager.startTransition()`	Crée une occurrence de transition, puis la lance.
`TransitionManager.toString()`	Renvoie le type de la classe TransitionManager sous la forme d'une chaîne.

Récapitulatif des propriétés de la classe TransitionManager

Le tableau suivant répertorie les propriétés de la classe TransitionManager.

Propriété	Description
`TransitionManager.content`	Occurrence de clip à laquelle TransitionManager doit appliquer une transition.
`TransitionManager.contentAppearance`	Objet qui contient les propriétés visuelles enregistrées du contenu (clip cible) auquel les transitions seront appliquées. Cette propriété est en lecture seule.

Récapitulatif des événements de la classe TransitionManager

Le tableau suivant répertorie les événements de la classe TransitionManager.

Evénement	Description
`TransitionManager.allTransitionsInDone`	Diffusé par une occurrence de la classe TransitionManager lorsqu'une transition ayant une propriété de direction `mx.transitions.Transition.IN` a été effectuée.
`TransitionManager.allTransitionsOutDone`	Diffusé par une occurrence de la classe TransitionManager lorsqu'une transition ayant une propriété de direction `mx.transitions.Transition.OUT` a été effectuée.

TransitionManager.allTransitionsInDone

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.allTransitionsInDone = function(eventObj:Object) {
  // ...
};
transitionManagerInstance.addEventListener("allTransitionsInDone",
  listenerObject);
```

Description

Evénement : indique aux écouteurs que l'occurrence de la classe TransitionManager a effectué toutes les transitions comportant une propriété de direction `mx.transitions.Transition.IN` et les a supprimées de la liste des transitions à appliquer.

Cet exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de la classe TransitionManager (*transitionManagerInstance*) distribue un événement (ici, `allTransitionsInDone`) qui est géré par une fonction (également appelée *gestionnaire*) sur un objet écouteur (`listenerObject`) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, elle transmet automatiquement un objet événement (`eventObject`) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Dans ce cas, l'événement `allTransitionsInDone` fournit une propriété cible qui contient l'occurrence TransitionManager qui a déclenché l'événement vous permettant d'utiliser cette dernière ainsi que toutes ses propriétés et méthodes dans le code qui reçoit l'événement `allTransitionsInDone`. Pour plus d'informations, voir Chapitre 21, « Classe EventDispatcher », à la page 515.

Exemple

L'exemple suivant affecte un objet pour écouter l'événement `allTransitionsInDone` et indique la méthode servant de gestionnaire pour l'événement. Lorsque cette méthode est appelée pour gérer l'événement, une transition comportant une propriété de direction `mx.transitions.Transition.IN` a déjà été effectuée.

```
import mx.transitions.*;
import mx.transitions.easing.*;
var myTransitionManager:TransitionManager = new TransitionManager(img1_mc);
myTransitionManager.startTransition({type:Iris, direction:Transition.IN,
  duration:1, easing:None.easeNone, startPoint:5, shape:Iris.CIRCLE});

var myListener:Object = new Object();
myListener.allTransitionsInDone = function(eventObj:Object) {
  trace("allTransitionsInDone event occurred.");
};
myTransitionManager.addEventListener("allTransitionsInDone", myListener);
```

Voir aussi

Chapitre 21, « Classe EventDispatcher », à la page 515

TransitionManager.allTransitionsOutDone

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.allTransitionsOutDone = function(eventObj:Object) {
  // ...
};
transitionManagerInstance.addEventListener("allTransitionsOutDone",
  listenerObject);
```

Description

Evénement ; indique aux écouteurs que l'occurrence de la classe TransitionManager a effectué toutes les transitions comportant une propriété de direction « out » et les a supprimées de la liste des transitions à appliquer.

Cet exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de la classe TransitionManager (`transitionManagerInstance`) distribue un événement (ici, `allTransitionsOutDone`) qui est géré par une fonction (également appelée *gestionnaire*) sur un objet écouteur (`listenerObject`) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, elle transmet automatiquement un objet événement (`eventObject`) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Dans ce cas, l'événement `allTransitionsOutDone` fournit une propriété cible qui contient l'occurrence TransitionManager qui a déclenché l'événement vous permettant d'utiliser cette dernière ainsi que toutes ses propriétés et méthodes dans le code qui reçoit l'événement `allTransitionsOutDone`. Pour plus d'informations, voir Chapitre 21, « Classe EventDispatcher », à la page 515.

Exemple

L'exemple suivant affecte un objet pour écouter l'événement `allTransitionsOutDone` et indique la méthode servant de gestionnaire pour l'événement. Lorsque cette méthode est appelée pour gérer l'événement, une transition comportant une propriété de direction `mx.transitions.Transition.OUT` a déjà été effectuée.

```
import mx.transitions.*;
import mx.transitions.easing.*;
var myTransitionManager:TransitionManager = new TransitionManager(img1_mc);
myTransitionManager.startTransition({type:Iris, direction:Transition.OUT,
  duration:1, easing:None.easeNone,startPoint:5, shape:Iris.CIRCLE});

var myListener:Object = new Object();
myListener.allTransitionsOutDone = function(eventObj:Object) {
  trace("allTransitionsOutDone event occurred.");
};
myTransitionManager.addEventListener("allTransitionsOutDone", myListener);
```

Voir aussi

Chapitre 21, « Classe EventDispatcher », à la page 515

TransitionManager.content

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`transitionManagerInstance``.content`

Description

Propriété : occurrence de clip à laquelle la classe TransitionManager doit appliquer une transition.

Exemple

L'exemple suivant renvoie l'objet clip actuellement ciblé par une occurrence TransitionManager :

```
import mx.transitions.*;
import mx.transitions.easing.*;
var myTransitionManager:TransitionManager = new TransitionManager(img1_mc);
var myMovieClip:MovieClip = myTransitionManager.content;
trace(myMovieClip._name);
```

TransitionManager.contentAppearance

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

transitionManagerInstance.contentAppearance

Description

Propriété (lecture seule) : objet contenant un instantané des propriétés du clip cible d'une occurrence TransitionManager avant l'application de la transition. Cet objet permet d'obtenir des informations sur les valeurs de propriété que peut renvoyer le clip à la fin de la transition. L'objet renvoyé par `TransitionManager.contentAppearance` contient un enregistrement des paramètres correspondants d'origine du clip cible dans les propriétés suivantes : `_x`, `_y`, `_xscale`, `_yscale`, `_alpha`, `_rotation`, `_innerBounds`, `_outerBounds`, `_width`, `_height` et `colorTransform`. Ces propriétés sont enregistrées, et la méthode `TransitionManager.start()` ou `TransitionManager.startTransition()` est appelée.

Exemple

L'exemple suivant appelle la méthode `TransitionManager.contentAppearance()` pour obtenir du clip cible de l'objet TransitionManager les paramètres de propriété d'origine avant l'application de la transition :

```
import mx.transitions.*;
import mx.transitions.easing.*;
var myTransitionManager:TransitionManager = new TransitionManager(img1_mc);
myTransitionManager.startTransition({type:Zoom, direction:Transition.OUT,
  duration:3, easing:Bounce.easeOut});

var myMovieClip:MovieClip = myTransitionManager.content;
var myOriginalMovieClipProps:Object =
  myTransitionManager.contentAppearance;

for (var prop in myOriginalMovieClipProps) {
 trace(myMovieClip._name + "." +prop+" = "+myOriginalMovieClipProps[prop]);
}
```

TransitionManager.start()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

transitionManagerInstance.start(*content*, *transParams*)

Paramètres

content Objet MovieClip auquel appliquer l'effet de transition.

transParams Ensemble des paramètres transmis dans un objet.

L'objet `transParams` doit contenir un paramètre `type` qui indique la classe d'effet de transition à appliquer, suivi des paramètres `direction`, `duration` et `easing`. Par ailleurs, vous devez inclure tous les paramètres requis par cette classe d'effet de transition. Par exemple, la classe d'effet de transition `mx.transitions.Iris` requiert les autres paramètres `startPoint` et `shape`. Ainsi, outre les paramètres `type`, `duration` et `easing` requis par chaque transition, vous devez ajouter (à l'objet `transParams`) les paramètres `startPoint` et `shape` que l'effet `mx.transitions.Iris` nécessite. Le code suivant ajoute les paramètres `startPoint` et `shape` à l'effet `mx.transitions.Iris` :

```
{type:mx.transitions.Iris, direction:mx.transitions.Transition.IN,
  duration:5, easing:mx.transitions.easing.Bounce.easeOut,
  startPoint:5, shape:mx.transitions.Iris.CIRCLE}
```

Pour vérifier les autres paramètres requis pour la classe d'effet de transition que vous indiquez dans le paramètre `type` de l'objet `transParam`, reportez-vous à l'API de cette classe.
Par exemple, pour plus d'informations sur la classe de transition Blinds, reportez-vous à « Transition Blinds », à la page 1299.

REMARQUE

Le paramètre `type` de l'objet `transParams` doit inclure le nom complet de paquet des classes spécifiées pour ses paramètres, à moins qu'elles soient déjà importées à l'aide de l'instruction `import`. Pour que vous ne soyez pas tenu d'indiquer le nom complet du paquet de classes pour l'ensemble des paramètres `transParams`, insérez d'abord les instructions `import` suivantes dans votre code afin d'importer toutes les classes `mx.transitions` et `mx.transitions.easing`.

```
import mx.transitions.*;
import mx.transitions.easing.*;
```

Valeur renvoyée

Une occurrence de l'objet Transition que l'occurrence de la classe TransitionManager doit appliquer.

Description

Méthode : crée une occurrence de la classe TransitionManager si elle n'existe pas, crée une occurrence de la classe de transition spécifiée dans le paramètre `transParams.type`, puis lance la transition. Cette transition est appliquée à la diapositive ou au clip désigné dans le paramètre `content`.

Exemple

Le code suivant utilise la méthode `TransitionManager.start()` pour créer une occurrence de la classe TransitionManager et affecte une transition Iris à un clip nommé img1_mc. La méthode `TransitionManager.start()` contient deux paramètres : le premier, `content`, correspond à l'objet MovieClip auquel l'effet de transition va s'appliquer et le second, `transParam`, contient un objet qui comprend un ensemble de paramètres. Cet objet, qui contient un ensemble de paramètres, désigne d'abord le type d'effet de transition avec le paramètre `type`, suivi des paramètres `direction`, `duration` et `easing`. Ces quatre paramètres constituent les informations requises pour tous les effets TransitionManager. Tous les paramètres requis de façon spécifique par le type de transition sont insérés après le paramètre `easing`. Dans l'exemple suivant, Iris correspond au type de transition ; la transition Iris requiert les paramètres `startPoint` et `shape` (pour plus d'informations sur les paramètres de transition Iris, reportez-vous à « Transition Iris », à la page 1301).

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start(img1_mc, {type:Iris, direction:Transition.IN,
  duration:5, easing:Bounce.easeOut, startPoint:5, shape:Iris.CIRCLE});
```

TransitionManager.startTransition()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`transitionManagerInstance``.startTransition(`*`transParams`*`)`

Paramètres

transParams Ensemble des paramètres transmis dans un objet.

L'objet `transParams` doit contenir un paramètre `type` qui indique la classe d'effet de transition à appliquer, suivi des paramètres `direction`, `duration` et `easing`. Par ailleurs, vous devez inclure tous les paramètres requis par la classe d'effet de transition spécifiée. Par exemple, la classe d'effet de transition `mx.transitions.Iris` requiert les autres paramètres `startPoint` et `shape`. Ainsi, outre les paramètres `type`, `duration` et `easing` requis par chaque transition, vous devez ajouter (à l'objet `transParams`) les paramètres `startPoint` et `shape` que l'effet `mx.transitions.Iris` nécessite. Le code suivant ajoute les paramètres `startPoint` et `shape` à l'effet `mx.transitions.Iris` :

```
{type:mx.transitions.Iris, direction:mx.transitions.Transition.IN,
  duration:5, easing:mx.transitions.easing.Bounce.easeOut, startPoint:5,
  shape:mx.transitions.Iris.CIRCLE}
```

Pour vérifier les autres paramètres requis pour la classe d'effet de transition que vous indiquez dans le paramètre `type` de l'objet `transParam`, reportez-vous à l'API de cette classe. Par exemple, pour plus d'informations sur la classe de transition Blinds, reportez-vous à « Transition Blinds », à la page 1299.

REMARQUE

Le paramètre type de l'objet `transParams` doit inclure le nom complet de paquet des classes spécifiées pour ses paramètres, à moins qu'elles soient déjà importées à l'aide de l'instruction `import`. Pour que vous ne soyez pas tenu d'indiquer le nom complet du paquet de classes pour l'ensemble des paramètres `transParams`, insérez d'abord les instructions `import` suivantes dans votre code afin d'importer toutes les classes `mx.transitions` et `mx.transitions.easing`.

```
import mx.transitions.*;
import mx.transitions.easing.*;
```

Valeur renvoyée

Une occurrence de l'objet Transition que l'occurrence de la classe TransitionManager doit appliquer.

Description

Méthode : crée et démarre une occurrence de la classe de transition spécifiée qui sera appliquée à la diapositive ou au clip affecté par l'occurrence TransitionManager. Si une transition compatible existe, elle est supprimée et une nouvelle transition est créée et démarrée.

Exemple

Le code suivant importe la classe TransitionManager et crée une occurrence TransitionManager. La méthode `TransitionManager.startTransition()` désigne ensuite une transition `mx.transitions.Zoom` dans son paramètre `type`. Le paramètre `direction` indique que la transition doit se déplacer dans la direction out en définissant `mx.transitions.Transition.OUT`. La durée de la transition est de 3 secondes. L'accélération est calculée à l'aide de la méthode `mx.transitions.Bounce.easeOut()` de la classe Bounce. Cet effet, qui dure au total 3 secondes, provoque l'affichage du clip img1_mc en zoom arrière dans un mouvement de rebond, jusqu'à sa disparition.

```
import mx.transitions.*;
import mx.transitions.easing.*;
var myTransitionManager:TransitionManager = new TransitionManager(img1_mc);
myTransitionManager.startTransition({type:Zoom, direction:Transition.OUT,
  duration:3, easing:Bounce.easeOut});
```

TransitionManager.toString()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`transitionManagerInstance``.toString()`

Valeur renvoyée

La chaîne suivante est renvoyée : `"[TransitionManager]"`.

Description

Méthode : renvoie le type de l'objet TransitionManager sous la forme d'une chaîne.

Exemple

Le code suivant indique à l'occurrence de la classe TransitionManager de renvoyer une chaîne indiquant son type :

```
import mx.transitions.*;
import mx.transitions.easing.*;
var myTransitionManager:TransitionManager = new TransitionManager(img1_mc);
var myType:String = myTransitionManager.toString();
trace(myType);
```

Classes basées sur la classe Transition

Héritage (classe Root)

Nom de classe ActionScript mx.transitions.Transition

La classe Transition est une classe de base pour toutes les classes de transition. Vous n'utilisez ni accédez à cette classe directement. Elle permet aux classes basées sur une classe Transition de partager des propriétés et des comportements communs auxquels accède une occurrence de la classe TransitionManager. Les classes basées sur la classe Transition définissent un effet qui est appliqué au fil du temps à un clip ou à une diapositive.

Flash inclut dix transitions qui vous permettent d'appliquer des effets à des objets clip.

Vous pouvez toutes les personnaliser en incluant des méthodes d'accélération facultatives, et la plupart acceptent plusieurs paramètres facultatifs qui vous permettent de contrôler des aspects particuliers de leur effet. L'*accélération* fait référence à l'accélération ou à la décélération progressive pendant une animation, qui lui donne un aspect plus réaliste. Par exemple, une balle peut prendre progressivement de la vitesse au début d'une animation, mais ralentir avant de s'arrêter complètement à la fin de l'animation. Il existe de nombreuses équations pour cette accélération et cette décélération, qui modifient l'animation d'accélération.

Les transitions sont utilisées avec la classe TransitionManager. Voir « Classe TransitionManager », à la page 1285. Vous pouvez utiliser la classe TransitionManager pour définir une transition et l'appliquer à un objet clip au lieu de l'appeler directement.
Par exemple, pour appliquer une transition Zoom au clip, img1_mc, vous devez définir la classe de transition Zoom comme paramètre type dans `TransitionManager.start()` :

```
mx.transitions.TransitionManager.start(myMovieClip_mc,
  {type:mx.transitions.Zoom, direction:mx.transitions.Transition.IN,
  duration:1, easing:mx.transitions.easing.Bounce.easeOut});
```

Flash comprend les transitions suivantes :

Transition	Description
Transition Blinds	Révèle l'objet clip à l'aide de rectangles apparaissant ou disparaissant.
Transition Fade	Réalise un fondu de l'objet clip en entrée ou en sortie.
Transition Fly	Insère l'objet clip depuis une direction particulière.
Transition Iris	Révèle ou masque l'objet clip à l'aide d'un masque animé ou d'un carré ou d'un cercle qui effectue un zoom avant ou arrière.
Transition Photo	Fait apparaître ou disparaître l'objet clip comme un flash de photo.
Transition PixelDissolve	Révèle ou masque l'objet clip à l'aide de rectangles, qui apparaissent ou disparaissent de façon aléatoire dans le motif en damier.

Transition	Description
Transition Rotate	Fait pivoter l'objet clip.
Transition Squeeze	Redimensionne l'objet clip horizontalement ou verticalement.
Transition Wipe	Révèle ou masque l'objet clip à l'aide d'un masque animé ou d'une forme qui se déplace horizontalement.
Transition Zoom	Effectue un zoom avant ou arrière sur l'objet clip en le redimensionnant proportionnellement.

Transition Blinds

Nom de classe ActionScript mx.transitions.Blinds

Paramètres

numStrips Nombre de bandes masquantes dans l'effet Blinds. La plage recommandée est comprise entre 1 et 50.

dimension Nombre entier indiquant si les bandes Blinds doivent être verticales (0) ou horizontales (1).

Description

Effet de transition : révèle l'objet clip à l'aide de rectangles apparaissant ou disparaissant.

Cette classe est utilisée en spécifiant `mx.transitions.Blinds` comme paramètre `transObject.type` de la classe TransitionManager.

Exemple

Le code suivant crée une occurrence TransitionManager qui applique la transition Blinds avec 10 `numStrips` et un nombre entier `dimension` défini comme vertical (0). Le contenu cible de la transition est le clip img1_mc. L'occurrence TransitionManager applique une direction `mx.transitions.Transition.IN` pendant 2 secondes sans accélération.

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start(img1_mc, {type:Blinds, direction:Transition.IN,
  duration:2, easing:None.easeNone, numStrips:10, dimension:0});
```

Transition Fade

Nom de classe ActionScript mx.transitions.Fade

Description

Effet de transition : réalise un fondu de l'objet clip en entrée ou en sortie.

Cette classe est utilisée en spécifiant `mx.transitions.Fade` comme paramètre `transObject.type` de la classe TransitionManager.

Exemple

Le code suivant crée une occurrence TransitionManager qui applique la transition Fade. Le contenu cible de la transition est le clip img1_mc. L'occurrence TransitionManager applique une direction `mx.transitions.Transition.IN` pendant 3 secondes sans accélération.

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start(img1_mc, {type:Fade, direction:Transition.IN,
  duration:3, easing:None.easeNone});
```

Transition Fly

Nom de classe ActionScript mx.transitions.Fly

Paramètres

`startPoint` Nombre entier, compris entre 1 et 9, qui indique une position de départ :

Haut gauche, 1 ; Haut centre, 2 ; Haut droit, 3 ; Centre gauche, 4 ; Centre, 5 ; Centre droit, 6 ; Bas gauche, 7 ; Bas centre, 8 ; Bas droit, 9.

Description

Effet de transition : insère l'objet clip depuis une direction particulière.

Cette classe est utilisée en spécifiant `mx.transitions.Fly` comme paramètre `transObject.type` de la classe TransitionManager.

Exemple

Le code suivant crée une occurrence TransitionManager qui applique la transition Fly, le paramètre `startPoint` étant défini sur 9 (bas droit). Le contenu cible de la transition est le clip img1_mc. L'occurrence TransitionManager applique une direction `mx.transitions.Transition.IN` pendant 3 secondes avec l'effet d'accélération `Elastic.easeOut`.

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start(img1_mc, {type:Fly, direction:Transition.IN,
  duration:3, easing:Elastic.easeOut, startPoint:9});
```

Transition Iris

Nom de classe ActionScript mx.transitions.Iris

Paramètres

`startPoint` Nombre entier, compris entre 1 et 9, qui indique une position de départ :

Haut gauche, 1 ; Haut centre, 2 ; Haut droit, 3 ; Centre gauche, 4 ; Centre, 5 ; Centre droit, 6 ; Bas gauche, 7 ; Bas centre, 8 ; Bas droit, 9.

`shape` Masque de forme carrée `mx.transitions.Iris.SQUARE` ou arrondie `mx.transitions.Iris.CIRCLE`.

Description

Effet de transition : révèle l'objet clip à l'aide d'un masque animé de forme carrée ou arrondie qui effectue un zoom avant ou arrière.

Cette classe est utilisée en spécifiant `mx.transitions.Iris` comme paramètre `transObject.type` de la classe TransitionManager.

Exemple

Le code suivant crée une occurrence TransitionManager qui applique la transition Iris avec le paramètre `startPoint` défini à partir du centre (5) et la forme masquante `mx.transitions.Iris.CIRCLE`. Le contenu cible de la transition est le clip img1_mc. La classe TransitionManager applique la direction `mx.transitions.Transition.IN` pendant 2 secondes avec une accélération `Strong` et une emphase `easeOut` en indiquant la méthode de calcul d'accélération `mx.transitions.easing.Strong.easeOut`.

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start(img1_mc, {type:Iris, direction:Transition.IN,
  duration:2, easing:Strong.easeOut, startPoint:5, shape:Iris.CIRCLE});
```

Transition Photo

Nom de classe ActionScript mx.transitions.Photo

Description

Effet de transition : fait apparaître ou disparaître l'objet clip tel un flash d'appareil photo.

Cette classe est utilisée en spécifiant `mx.transitions.Photo` comme paramètre `transObject.type` de la classe TransitionManager.

Exemple

Le code suivant crée une occurrence TransitionManager qui applique la transition Photo au contenu cible, le clip img1_mc. L'occurrence TransitionManager applique la direction `mx.transitions.Transition.IN` pendant 1 seconde sans accélération.

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start (img1_mc, {type:Photo, direction:Transition.IN,
  duration:1, easing:None.easeNone});
```

Transition PixelDissolve

Nom de classe ActionScript mx.transitions.PixelDissolve

Paramètres

`xSections` Nombre entier qui indique le nombre de sections de rectangles masquants le long de l'axe horizontal. La plage recommandée est comprise entre 1 et 50.

`ySections` Nombre entier qui indique le nombre de sections de rectangles masquants le long de l'axe vertical. La plage recommandée est comprise entre 1 et 50.

Description

Effet de transition : révèle l'objet clip à l'aide de rectangles qui apparaissent ou disparaissent de façon aléatoire dans le motif en damier.

Cette classe est utilisée en spécifiant `mx.transitions.PixelDissolve` comme paramètre `transObject.type` de la classe TransitionManager.

Exemple

Le code suivant crée une occurrence TransitionManager qui applique la transition PixelDissolve avec 10 `xSections` et 10 `ySections`. Le contenu cible de la transition est le clip img1_mc. L'occurrence TransitionManager applique une direction `mx.transitions.Transition.IN` pendant 2 secondes sans accélération.

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start(img1_mc, {type:PixelDissolve,
  direction:Transition.IN, duration:2, easing:None.easeNone, xSections:10,
  ySections:10});
```

Transition Rotate

Nom de classe ActionScript mx.transitions.Rotate

Paramètres

`ccw` Valeur booléenne : *`false`* pour une rotation vers la droite ; *`true`* pour une rotation vers la gauche.

`degrees` Nombre entier indiquant de degré de la rotation de l'objet. La plage recommandée est comprise entre 1 et 9999. Par exemple, si vous définissez le paramètre `degrees` sur 1080, l'objet tourne complètement sur lui trois fois.

Description

Effet de transition : fait pivoter l'objet clip.

Cette classe est utilisée en spécifiant `mx.transitions.Rotate` comme paramètre `transObject.type` de la classe TransitionManager.

Exemple

Le code suivant crée une occurrence TransitionManager qui applique la transition Rotate, dans le sens horaire sur 720 degrés (deux tours complets). Le contenu cible de la transition est le clip img1_mc. L'occurrence TransitionManager applique la direction `mx.transitions.Transition.IN` pendant 3 secondes avec une accélération définie sur `Strong.easeInOut` afin que la transition démarre lentement, accélère, puis se termine lentement.

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start(img1_mc, {type:Rotate, direction:Transition.IN,
  duration:3, easing:Strong.easeInOut, ccw:false, degrees:720});
```

Transition Squeeze

Nom de classe ActionScript mx.transitions.Squeeze

Paramètres

`dimension` Nombre entier indiquant que l'effet Squeeze doit être vertical (1) ou horizontal (0).

Description

Effet de transition : redimensionne l'objet clip horizontalement ou verticalement.

Cette classe est utilisée en spécifiant `mx.transitions.Squeeze` comme paramètre `transObject.type` de la classe TransitionManager.

Exemple

Le code suivant crée une occurrence TransitionManager qui applique la transition Squeeze avec un nombre entier `dimension` défini sur vertical (1). Le contenu cible de la transition est le clip img1_mc. La classe TransitionManager applique la direction `mx.transitions.Transition.IN` pendant 2 secondes avec un effet d'accélération `Elastic` dans la direction `easeOut`.

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start(img1_mc, {type:Squeeze, direction:Transition.IN,
  duration:2, easing:Elastic.easeOut, dimension:1});
```

Transition Wipe

Nom de classe ActionScript mx.transitions.Wipe

Paramètres

`startPoint` Nombre entier indiquant une position de départ. Plage comprise de 1 à 4 et de 6 à 9 :

Haut gauche, 1 ; Haut centre, 2 ; Haut droit, 3 ; Centre gauche, 4 ; Centre droit, 6 ; Bas gauche, 7 ; Bas centre, 8 ; Bas droit, 9.

Description

Effet de transition : révèle ou masque l'objet clip à l'aide d'un masque animé ou d'une forme qui se déplace horizontalement.

Cette classe est utilisée en spécifiant `mx.transitions.Wipe` comme paramètre `transObject.type` de la classe TransitionManager.

Exemple

Le code suivant crée une occurrence TransitionManager qui applique la transition Wipe avec le paramètre `startPoint` défini sur 1 (haut gauche). Le contenu cible de la transition est le clip img1_mc. L'occurrence TransitionManager applique la direction `mx.transitions.Transition.IN` pendant 2 secondes sans accélération.

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start(img1_mc, {type:Wipe, direction:Transition.IN,
  duration:2, easing:None.easeNone, startPoint:1});
```

Transition Zoom

Nom de classe ActionScript mx.transitions.Zoom

Description

Effet de transition : effectue un zoom avant ou arrière sur l'objet clip en le redimensionnant proportionnellement.

Cette classe est utilisée en spécifiant `mx.transitions.Zoom` comme paramètre `transObject.type` de la classe TransitionManager.

Exemple

Le code suivant crée une occurrence TransitionManager qui applique la transition Zoom au contenu cible, le clip img1_mc. La classe TransitionManager applique la direction `mx.transitions.Transition.IN` pendant 2 secondes, avec une accélération de type `Elastic` qui commence rapidement et accélère lentement sur la fin.

```
import mx.transitions.*;
import mx.transitions.easing.*;
TransitionManager.start(img1_mc, {type:Zoom, direction:Transition.IN,
  duration:2, easing:Elastic.easeOut});
```

CHAPITRE 49

Interface TreeDataProvider

49

L'interface TreeDataProvider est un ensemble de propriétés et de méthodes qui n'a pas besoin d'être instancié pour être utilisé. Si une classe Tree est comprise dans un fichier SWF, toutes les occurrences XML de ce fichier contiennent l'interface TreeDataProvider. Les nœuds d'une arborescence sont autant d'objets XML contenant l'interface TreeDataProvider.

REMARQUE

L'interface TreeDataProvider est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Il est recommandé d'utiliser les méthodes de TreeDataProvider pour créer du code XML pour la propriété `Tree.dataProvider`, car seule l'interface TreeDataProvider diffuse des événements qui actualisent l'affichage de l'arborescence. Il s'agit d'événements gérés par la classe Tree ; vous ne devez pas écrire de fonctions pour gérer ces événements. (Les méthodes de la classe XML intégrée ne diffusent pas de tels événements.)

Utilisez les méthodes de TreeDataProvider pour contrôler le modèle et l'affichage des données. Utilisez les méthodes de la classe XML intégrée pour les tâches en lecture seule (parcourir la hiérarchie de l'arborescence, par exemple).

Vous pouvez sélectionner la propriété qui contrôle le texte à afficher en spécifiant une propriété `labelField` ou `labelFunction`. Par exemple, le code `myTree.labelField = "firstName";` renvoie une requête sur la valeur de la propriété `myTreeDP.attributes.fred` pour le texte d'affichage.

Méthodes de l'interface TreeDataProvider

Le tableau suivant répertorie les méthodes de l'interface TreeDataProvider.

Méthode	Description
`TreeDataProvider.addTreeNode()`	Ajoute un nœud enfant à la racine de l'arborescence.
`TreeDataProvider.addTreeNodeAt()`	Ajoute un nœud enfant à un endroit spécifié sur le nœud parent.
`TreeDataProvider.getTreeNodeAt()`	Renvoie l'enfant spécifié d'un nœud.
`TreeDataProvider.removeTreeNode()`	Supprime un nœud et tous les descendants du nœud à partir du parent du nœud.
`TreeDataProvider.removeTreeNodeAt()`	Supprime un nœud et tous les descendants du nœud à partir de la position dans l'index du nœud enfant.

Propriétés de l'interface TreeDataProvider

Le tableau suivant répertorie les propriétés de l'interface TreeDataProvider.

Propriété	Description
`TreeDataProvider.attributes.data`	Spécifie les données à associer à un nœud.
`TreeDataProvider.attributes.label`	Spécifie le texte à afficher à côté d'un nœud.

TreeDataProvider.addTreeNode()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
someNode.addTreeNode(label, data)
```

Utilisation 2 :

```
someNode.addTreeNode(child)
```

Paramètres

label Chaîne qui affiche le nœud.

data Objet de n'importe quel type associé au nœud.

child Tout objet XMLNode.

Valeur renvoyée

Le nœud XML ajouté.

Description

Méthode : ajoute un nœud enfant à la racine de l'arborescence. Le nœud est construit à partir des informations fournies dans les paramètres *label* et *data* (Utilisation 1) ou à partir du nœud enfant (objet XMLNode) prédéfini (Utilisation 2). Si le nœud ajouté existe déjà, il est supprimé de son emplacement précédent.

L'appel à cette méthode actualise l'affichage de l'arborescence.

Exemple

La première ligne du code dans l'exemple suivant localise le nœud auquel ajouter un enfant. La deuxième ligne ajoute un nouveau nœud à un nœud spécifié.

```
var myTreeNode = myTreeDP.firstChild.firstChild;
myTreeNode.addTreeNode("Inbox", 3);
```

Le code suivant déplace un nœud d'une arborescence vers la racine d'une autre arborescence :

```
myTreeNode.addTreeNode(mySecondTree.getTreeNodeAt(3));
```

TreeDataProvider.addTreeNodeAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
someNode.addTreeNodeAt(index, label, data)
```

Utilisation 2 :

```
someNode.addTreeNodeAt(index, child)
```

Paramètres

index Nombre entier indiquant la position d'index (parmi les nœuds enfants) à laquelle le nœud doit être ajouté.

label Chaîne qui affiche le nœud.

data Objet de n'importe quel type associé au nœud.

child Tout objet XMLNode.

Valeur renvoyée

Le nœud XML ajouté.

Description

Méthode : ajoute un nœud enfant à l'endroit spécifié dans le nœud parent. Le nœud est construit à partir des informations fournies dans les paramètres *label* et *data* (Utilisation 1) ou à partir du nœud enfant (objet XMLNode) prédéfini (Utilisation 2). Si le nœud ajouté existe déjà, il est supprimé de son emplacement précédent.

L'appel à cette méthode actualise l'affichage de l'arborescence.

Exemple

Le code suivant localise le nœud auquel vous ajoutez un nœud et en ajoute un nouveau en tant que deuxième enfant de la racine :

```
var myTreeNode = myTreeDP.firstChild.firstChild;
myTreeNode.addTreeNodeAt(1, "Inbox", 3);
```

Le code suivant déplace un nœud à partir d'une arborescence pour qu'il devienne le quatrième enfant de la racine d'une autre arborescence :

```
myTreeNode.addTreeNodeAt(3, mySecondTree.getTreeNodeAt(3));
```

TreeDataProvider.attributes.data

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
someNode.attributes.data
```

Description

Propriété : spécifie les données à associer au nœud. Ceci permet d'ajouter la valeur en tant qu'attribut dans l'objet XML. La définition de cette propriété n'actualise pas l'affichage. Cette propriété peut être de n'importe quel type de données.

Exemple

Le code suivant localise le nœud à régler et définit sa propriété `data` :

```
var myTreeNode = myTreeDP.firstChild.firstChild;
myTreeNode.attributes.data = "hi"; // résulte en <node data = "hi">;
```

Voir aussi

`TreeDataProvider.attributes.label`

TreeDataProvider.attributes.label

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`someNode``.attributes.label`

Description

Propriété : chaîne indiquant le texte affiché pour le nœud. Il est affiché dans un attribut de l'objet XMLNode. La définition de cette propriété n'actualise pas l'affichage.

Exemple

Le code suivant localise le nœud à régler et définit sa propriété `label`. Le résultat du code suivant est `<node label="Mail">`.

```
var myTreeNode = myTreeDP.firstChild.firstChild;
myTreeNode.attributes.label = "Mail";
```

Voir aussi

`TreeDataProvider.attributes.data`

TreeDataProvider.getTreeNodeAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
someNode.getTreeNodeAt(index)
```

Paramètres

index Nombre entier représentant la position du nœud enfant dans le nœud actuel.

Valeur renvoyée

Le nœud spécifié.

Description

Méthode : renvoie le nœud enfant spécifié du nœud.

Exemple

Le code suivant localise un nœud, puis récupère le deuxième enfant de `myTreeNode` :

```
var myTreeNode = myTreeDP.firstChild.firstChild;
myTreeNode.getTreeNodeAt(1);
```

TreeDataProvider.removeTreeNode()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
someNode.removeTreeNode()
```

Paramètres

Aucun.

Valeur renvoyée

Le nœud XML supprimé ou `undefined` en cas d'erreur.

Description

Méthode : supprime le nœud spécifié et ses éventuels descendants, à partir de son parent.

Exemple

Le code suivant supprime un nœud :

```
myTreeDP.firstChild.removeTreeNode();
```

TreeDataProvider.removeTreeNodeAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`someNode``.removeTreeNodeAt(`*`index`*`)`

Paramètres

`index` Nombre entier indiquant la position du nœud à supprimer.

Valeur renvoyée

Le nœud XML supprimé ou `undefined` en cas d'erreur.

Description

Méthode : supprime un nœud (et tous les descendants) spécifié par le nœud actuel et la position d'index du nœud enfant. L'appel de cette méthode actualise l'affichage.

Exemple

Le code suivant supprime le quatrième enfant du nœud `myTreeDP.firstChild` :

```
myTreeDP.firstChild.removeTreeNodeAt(3);
```

CHAPITRE 50

Composant Tree

50

Le composant Tree permet à l'utilisateur d'afficher des données hiérarchiques. L'arborescence apparaît dans une zone (une occurrence du composant List, par exemple) ; chaque élément de l'arborescence, appelé *nœud* peut être une *feuille* ou une *branche*. Par défaut, une feuille est représentée par une étiquette de texte placée en regard d'une icône de fichier, tandis qu'une branche est représentée par une étiquette de texte placée en regard d'une icône de dossier, avec une flèche d'agrandissement (triangle d'affichage) que l'utilisateur peut ouvrir pour afficher les nœuds enfants. Les enfants d'une branche peuvent être des feuilles ou des branches.

REMARQUE

Le composant Tree est pris en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Les données d'un composant Tree doivent être fournies à partir d'une source de données XML. Pour plus d'informations, voir « Utilisation du composant Tree », à la page 1316.

Lorsqu'une occurrence de Tree a le focus (après que l'utilisateur a cliqué ou utilisé la touche de tabulation), vous pouvez utiliser les touches suivantes pour la contrôler :

Touche	Description
Flèche vers le bas	Déplace la sélection d'un élément vers le bas.
Flèche vers le haut	Déplace la sélection d'un élément vers le haut.
Flèche droite	Ouvre un nœud de branche sélectionné. Si une branche est déjà ouverte, se déplace vers le premier nœud enfant.
Flèche gauche	Ferme un nœud de branche sélectionné. Lorsque vous vous trouvez sur une feuille d'un nœud de branche fermé, se déplace vers le nœud parent.
Espace	Ouvre ou ferme un nœud de branche sélectionné.
Fin	Déplace la sélection en bas de la liste.

Touche	Description
Page d'accueil	Déplace la sélection en haut de la liste.
Pg. Suiv.	Déplace la sélection d'une page vers le bas.
Pg. Préc.	Déplace la sélection d'une page vers le haut.
Contrôle	Permet plusieurs sélections non contiguës.
Maj	Permet plusieurs sélections contiguës.

Le composant Tree n'est pas accessible dans les logiciels de lecture d'écran.

Utilisation du composant Tree

Le composant Tree permet de représenter des données hiérarchiques, telles que des dossiers d'adresses électroniques de clients, des fenêtres d'exploration de fichiers ou des systèmes de navigation par catégorie destinés aux inventaires. Le plus souvent, les données d'une arborescence sont récupérées à partir d'un serveur sous forme de code XML, mais il peut également s'agir de code XML créé pendant la programmation dans Flash. Le meilleur moyen de créer du code XML pour l'arborescence consiste à utiliser l'interface TreeDataProvider. Vous pouvez aussi utiliser la classe XML d'ActionScript ou créer une chaîne XML. Une fois votre source de données XML créée (ou chargée à partir d'une source externe), vous l'affectez à la propriété `Tree.dataProvider`.

Le composant Tree comprend deux ensembles d'API : la classe Tree et l'interface TreeDataProvider. La classe Tree contient les méthodes et propriétés de configuration visuelle. L'interface TreeDataProvider vous permet de construire votre code XML pour l'ajouter à plusieurs occurrences de composants Tree. Un objet TreeDataProvider diffuse des modifications vers l'ensemble des arborescences (occurrences de composant Tree) qui l'utilisent. De plus, tout objet XML ou XMLNode existant sur la même image en tant qu'arborescence ou menu reçoit automatiquement les méthodes et propriétés TreeDataProvider. Pour plus d'informations, voir « Interface TreeDataProvider », à la page 1307.

Formatage du code XML pour le composant Tree

Le composant Tree est conçu pour afficher des structures de données hiérarchiques en utilisant XML comme modèle de données. Il est important de comprendre la relation liant la source de données XML au composant Tree.

Examinez l'exemple de source de données XML suivant :

```
<node>
  <node label="Mail">
       <node label="INBOX"/>
       <node label="Personal Folder">
           <node label="Business" isBranch="true" />
           <node label="Demo" isBranch="true" />
           <node label="Personal" isBranch="true" />
           <node label="Saved Mail" isBranch="true" />
           <node label="bar" isBranch="true" />
       </node>
       <node label="Sent" isBranch="true" />
       <node label="Trash"/>
  </node>
</node>
```

REMARQUE

L'attribut `isBranch` est en lecture seule ; vous ne pouvez pas le définir directement. Pour le définir, appelez `Tree.setIsBranch()`.

Les nœuds de la source de données XML peuvent avoir n'importe quel nom. Notez dans l'exemple ci-dessus que chaque nœud porte l'appellation générique `node`. L'arborescence lit le fichier XML et construit la l'affichage de la structure hiérarchique en fonction de la relation imbriquée des nœuds.

Un nœud XML peut être de l'un des deux types suivants : branche ou feuille. Une branche peut contenir plusieurs nœuds enfants ; elle prend la forme d'une icône de dossier, avec une flèche d'agrandissement permettant à l'utilisateur d'ouvrir et de refermer le dossier. Une feuille prend la forme d'une icône de fichier ; elle ne peut pas contenir de nœuds enfants. Les feuilles et les branches peuvent être des racines ; un nœud de racine apparaît en haut de l'arborescence et n'a aucun parent. Les icônes sont personnalisables ; pour plus d'informations, reportez-vous à « Utilisation d'enveloppes avec le composant Tree », à la page 1330.

Il existe de nombreuses façons de structurer le fichier XML, mais le composant Tree ne peut pas utiliser tous les types de structures XML. N'imbriquez pas des attributs de nœud dans un nœud enfant ; chaque nœud doit contenir l'ensemble des attributs qui lui sont nécessaires. De même, les attributs de chaque nœud doivent être cohérents. Par exemple, pour décrire une structure de boîte de réception à l'aide d'un composant Tree, veillez à utiliser les mêmes attributs sur chaque nœud (message, données, heure, pièces jointes, etc.). Cela permet de définir les différents éléments pouvant être affichés ; vous pouvez explorer la hiérarchie pour comparer les données.

Lorsqu'une arborescence affiche un nœud, elle utilise l'attribut `label` du nœud par défaut comme étiquette de texte. S'il existe d'autres attributs, ils deviennent des propriétés complémentaires des attributs du nœud dans l'arborescence.

Le nœud racine réel est interprété comme étant le composant Tree lui-même. Cela signifie que le premier enfant (dans l'exemple ci-dessus, `<node label="Mail">`) devient le nœud racine dans l'arborescence. Ainsi une arborescence peut avoir plusieurs dossiers racines.
Dans l'exemple, un seul dossier racine est affiché dans l'arborescence : Mail. En revanche, si vous ajoutez des nœuds frère à ce niveau dans le fichier XML, plusieurs nœuds racines s'afficheront dans l'arborescence.

Un fournisseur de données pour une arborescence exige toujours d'avoir un nœud ayant des enfants qu'il peut afficher. Il affiche le premier enfant de l'objet XMLNode. Lorsque le fichier XML est enveloppé dans un objet XML, la structure a l'aspect suivant :

```
<XMLDocumentObject>
  <node>
    <node label="Mail">
      <node label="INBOX"/>
      <node label="Personal Folder">
        <node label="Business" isBranch="true" />
        <node label="Demo" isBranch="true" />
        <node label="Personal" isBranch="true" />
        <node label="Saved Mail" isBranch="true" />
        <node label="bar" isBranch="true" />
      </node>
      <node label="Sent" isBranch="true" />
      <node label="Trash"/>
    </node>
  </node>
</XMLDocumentObject>
```

Flash Player enveloppe les nœuds XML dans un nœud de document supplémentaire qui est transmis à l'arborescence. Lorsque l'arborescence tente d'afficher le fichier XML, elle tente d'afficher `<node>`, qui n'a pas d'étiquette, c'est pourquoi il ne s'affiche pas correctement.

Pour éviter ce problème, le fournisseur de données pour le composant Tree doit pointer vers le premier enfant de XMLDocumentObject, comme indiqué ici :

```
myTree.dataProvider = myXML.firstChild;
```

Paramètres de Tree

Vous pouvez définir les paramètres de création suivants pour chaque occurrence Tree dans l'inspecteur des propriétés ou l'inspecteur de composants :

multipleSelection est une valeur booléenne qui indique si l'utilisateur peut sélectionner plusieurs éléments (`true`) ou non (`false`). La valeur par défaut est `false`.

rowHeight indique la hauteur de chaque ligne, en pixels. La valeur par défaut est 20.

Vous pouvez rédiger du code ActionScript pour contrôler ces options et des options complémentaires pour le composant Tree à l'aide de ses propriétés, méthodes et événements. Pour plus d'informations, voir « Classe Tree », à la page 1330.

Vous ne pouvez pas entrer de paramètres de données dans l'inspecteur des propriétés ou dans l'inspecteur de composants du composant Tree comme vous le faites pour les autres composants. Pour plus d'informations, voir les sections « Utilisation du composant Tree », à la page 1316 et « Création d'une application à l'aide du composant Tree », à la page 1319.

Création d'une application à l'aide du composant Tree

Les procédures suivantes indiquent comment utiliser un composant Tree pour afficher des boîtes de réception dans une application de courrier électronique.

Le composant Tree ne vous permet pas d'entrer des paramètres de données dans l'inspecteur des propriétés ou l'inspecteur des composants. En raison de la complexité de la structure de données d'un composant Tree, vous devez soit importer un objet XML lors de l'exécution, soit en créer un dans Flash pendant la programmation. Pour créer un objet XML dans Flash, vous pouvez utiliser l'interface TreeDataProvider, l'objet XML d'ActionScript ou créer une chaîne XML. Chacune de ces options est expliquée dans les procédures suivantes.

Pour ajouter un composant Tree à une application et charger le fichier XML :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Enregistrez le document sous le nom **treeMenu.fla**.
3. Dans le panneau Composants, double-cliquez sur le composant Tree pour l'ajouter à la scène.
4. Sélectionnez l'occurrence Tree. Dans l'inspecteur des propriétés, entrez le nom d'occurrence **menuTree**.
5. Sélectionnez l'instance de Tree et appuyez sur F8. Sélectionnez Clip et entrez le nom **TreeNavMenu**.
6. Si l'écran des paramètres avancés n'est pas affiché, cliquez sur le bouton Avancé.

7. Sélectionnez Exporter pour ActionScript.
8. Tapez **TreeNavMenu** dans la zone de texte Classe, puis cliquez sur OK.
9. Choisissez Fichier > Nouveau et sélectionnez Fichier ActionScript.
10. Enregistrez le fichier sous le nom **TreeNavMenu.as** dans le même répertoire que treeMenu.fla.
11. Dans la fenêtre de script, entrez le code suivant :

```
import mx.controls.Tree;

class TreeNavMenu extends MovieClip {
  var menuXML:XML;
  var menuTree:Tree;
  function TreeNavMenu() {
    // Configuration de l'aspect de l'arborescence et des
    // gestionnaires d'événement.
    menuTree.setStyle("fontFamily", "_sans");
    menuTree.setStyle("fontSize", 12);
    // Chargement du menu XML.
    var treeNavMenu = this;
    menuXML = new XML();
    menuXML.ignoreWhite = true;
    menuXML.load("TreeNavMenu.xml");
    menuXML.onLoad = function() {
      treeNavMenu.onMenuLoaded();
    };
  }
  function change(event:Object) {
    if (menuTree == event.target) {
      var node = menuTree.selectedItem;
      // Développement/réduction, s'il s'agit d'une branche.
      if (menuTree.getIsBranch(node)) {
        menuTree.setIsOpen(node, !menuTree.getIsOpen(node), true);
      }
      // Accès, s'il s'agit d'un hyperlien.
      var url = node.attributes.url;
      if (url) {
        getURL(url, "_top");
      }
      // Effacement de toute sélection.
      menuTree.selectedNode = null;
    }
  }
  function onMenuLoaded() {
    menuTree.dataProvider = menuXML.firstChild;
    menuTree.addEventListener("change", this);
  }
}
```

Cet ActionScript configure des styles pour l'arborescence. Un objet XML est créé pour charger le fichier XML qui crée les nœuds de l'arborescence. Ensuite, le gestionnaire d'événements `onLoad` est défini pour configurer le fournisseur de données sur le contenu du fichier XML.

12. Créez un fichier appelé TreeNavMenu.xml dans un éditeur de texte.
13. Entrez le code suivant dans le fichier :

```
<node>
  <node label="My Bookmarks">
    <node label="Adobe Web site" url="http://www.adobe.com" />
    <node label="MXNA blog aggregator" url="http://www.markme.com/mxna"
  />
  </node>
  <node label="Google" url="http://www.google.com" />
</node>
```

14. Enregistrez vos documents et revenez au fichier treeMenu.fla. Choisissez Contrôle > Tester l'animation pour contrôler l'application.

Pour charger XML d'un fichier externe :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser une occurrence du composant Tree sur la scène.
3. Sélectionnez l'occurrence Tree. Dans l'inspecteur des propriétés, nommez l'occurrence **myTree**.
4. Dans le panneau Actions, ajoutez le code suivant à l'image 1 :

```
var myTreeDP:XML = new XML();
myTreeDP.ignoreWhite = true;
myTreeDP.load("treeXML.xml");
myTreeDP.onLoad = function() {
  myTree.dataProvider = this.firstChild;
};
var treeListener:Object = new Object();
treeListener.change = function(evt:Object) {
  trace("selected node is: "+evt.target.selectedNode);
  trace("");
};
myTree.addEventListener("change", treeListener);
```

Ce code crée un objet XML appelé `myTreeDP` et appelle la méthode `XML.load()` pour charger une source de données XML. Le code définit alors un gestionnaire d'événements `onLoad` qui définit la propriété `dataProvider` de l'occurrence `myTree` sur le nouvel objet XML pendant le chargement XML. Pour plus d'informations sur l'objet XML, consultez son entrée dans le *Guide de référence du langage ActionScript 2.0*.

5. Créez un fichier appelé **treeXML.xml** dans un éditeur de texte.

6. Entrez le code suivant dans le fichier :

```
<node>
  <node label="Mail">
    <node label="INBOX"/>
    <node label="Personal Folder">
      <node label="Business" isBranch="true" />
      <node label="Demo" isBranch="true" />
      <node label="Personal" isBranch="true" />
      <node label="Saved Mail" isBranch="true" />
      <node label="bar" isBranch="true" />
    </node>
    <node label="Sent" isBranch="true" />
    <node label="Trash"/>
  </node>
</node>
```

7. Choisissez Contrôle > Tester l'animation.

   Dans le fichier SWF, vous pouvez voir la structure XML qui s'affiche dans l'arborescence. Cliquez sur les éléments de l'arborescence pour voir les instructions `trace()` dans le gestionnaire d'événements `change` envoyer les données dans le panneau Sortie.

Pour utiliser la classe TreeDataProvider pour créer du code XML dans Flash pendant la création :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser une occurrence du composant Tree sur la scène.
3. Sélectionnez l'occurrence Tree et, dans l'inspecteur des propriétés, nommez l'occurrence **myTree**.
4. Dans le panneau Actions, ajoutez le code suivant à l'image 1 :

```
var myTreeDP:XML = new XML();
myTreeDP.addTreeNode("Local Folders", 0);
// Utilisez XML.firstChild pour imbriquer les nœuds enfants sous
// les dossiers locaux.
var myTreeNode:XMLNode = myTreeDP.firstChild;
myTreeNode.addTreeNode("Inbox", 1);
myTreeNode.addTreeNode("Outbox", 2);
myTreeNode.addTreeNode("Sent Items", 3);
myTreeNode.addTreeNode("Deleted Items", 4);
// Affectation de la source de données myTreeDP au composant myTree.
myTree.dataProvider = myTreeDP;
// Définition de chacun des quatre nœuds enfants comme étant des
// branches.
for (var i = 0; i<myTreeNode.childNodes.length; i++) {
  var node:XMLNode = myTreeNode.getTreeNodeAt(i);
  myTree.setIsBranch(node, true);
}
```

Ce code crée un objet XML appelé `myTreeDP`. Tout objet XML placé sur la même image qu'un composant Tree reçoit automatiquement l'ensemble des propriétés et des méthodes de l'interface TreeDataProvider. La deuxième ligne de code crée un nœud racine simple appelé Répertoires locaux. Pour plus d'informations sur le reste du code, consultez les commentaires (lignes précédées du signe `//`) dans le code.

5. Choisissez Contrôle > Tester l'animation.

   Dans le fichier SWF, vous pouvez voir la structure XML qui s'affiche dans l'arborescence. Cliquez sur les éléments de l'arborescence pour voir les instructions `trace()` dans le gestionnaire d'événements `change` envoyer les données dans le panneau Sortie.

Pour utiliser la classe XML ActionScript pour créer du code XML :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser une occurrence du composant Tree sur la scène.
3. Sélectionnez l'occurrence Tree. Dans l'inspecteur des propriétés, nommez l'occurrence, **myTree**.
4. Dans le panneau Actions, ajoutez le code suivant à l'image 1 :

```
// Création d'un objet XML
var myTreeDP:XML = new XML();
// Création des valeurs de nœud.
var myNode0:XMLNode = myTreeDP.createElement("node");
myNode0.attributes.label = "Local Folders";
myNode0.attributes.data = 0;
var myNode1:XMLNode = myTreeDP.createElement("node");
myNode1.attributes.label = "Inbox";
myNode1.attributes.data = 1;
var myNode2:XMLNode = myTreeDP.createElement("node");
myNode2.attributes.label = "Outbox";
myNode2.attributes.data = 2;
var myNode3:XMLNode = myTreeDP.createElement("node");
myNode3.attributes.label = "Sent Items";
myNode3.attributes.data = 3;
var myNode4:XMLNode = myTreeDP.createElement("node");
myNode4.attributes.label = "Deleted Items";
myNode4.attributes.data = 4;
// Affectation des nœuds à la hiérarchie de l'arborescence XML.
myTreeDP.appendChild(myNode0);
myTreeDP.firstChild.appendChild(myNode1);
myTreeDP.firstChild.appendChild(myNode2);
myTreeDP.firstChild.appendChild(myNode3);
myTreeDP.firstChild.appendChild(myNode4);
// Affectation de la source de données myTreeDP au composant Tree.
myTree.dataProvider = myTreeDP;
```

   Pour plus d'informations sur l'objet XML, consultez son entrée dans le *Guide de référence du langage ActionScript 2.0*.

5. Sélectionnez Contrôle > Tester l'animation.

   Dans le fichier SWF, vous pouvez voir la structure XML qui s'affiche dans l'arborescence. Cliquez sur les éléments de l'arborescence pour voir les instructions `trace()` dans le gestionnaire d'événements `change` envoyer les données dans le panneau Sortie.

Pour utiliser une chaîne formée pour créer du code XML dans Flash pendant la création :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser une occurrence du composant Tree sur la scène.
3. Sélectionnez l'occurrence Tree. Dans l'inspecteur des propriétés, nommez l'occurrence, **myTree**.
4. Dans le panneau Actions, ajoutez le code suivant à l'image 1 :

```
var myTreeDP:XML = new XML("<node label='Local Folders'><node
  label='Inbox' data='0'/><node label='Outbox' data='1'/></node>");
myTree.dataProvider = myTreeDP;
```

   Ce code crée l'objet XML `myTreeDP` et l'affecte à la propriété `dataProvider` de `myTree`.

5. Sélectionnez Contrôle > Tester l'animation.

   Dans le fichier SWF, vous pouvez voir la structure XML qui s'affiche dans l'arborescence. Cliquez sur les éléments de l'arborescence pour voir les instructions `trace()` dans le gestionnaire d'événements `change` envoyer les données dans le panneau Sortie.

Personnalisation du composant Tree

Vous pouvez transformer un composant Tree horizontalement et verticalement pendant la programmation et lors de l'exécution. Lors de la programmation, sélectionnez le composant sur la scène et utilisez l'outil Transformer librement ou une commande Modification > Transformer. A l'exécution, utilisez la méthode `setSize()` (voir `UIObject.setSize()`). Lorsqu'une arborescence n'est pas assez large pour afficher le texte des nœuds, celui-ci apparaît tronqué.

Utilisation de styles avec le composant Tree

Un composant Tree utilise les styles suivants :

Style	Thème	Description
`themeColor`	Halo	Couleur de base d'un composant. `"haloGreen"`, `"haloBlue"` et `"haloOrange"` sont des valeurs possibles. La valeur par défaut est `"haloGreen"`.
`backgroundColor`	Les deux	Couleur d'arrière-plan de la liste. La couleur par défaut est le blanc et est définie sur la déclaration de style de classe. Ce style est ignoré si `alternatingRowColors` est spécifié.
`backgroundDisabledColor`	Les deux	Couleur d'arrière-plan lorsque la propriété `enabled` du composant est définie sur `"false"`. La valeur par défaut est 0xDDDDDD (gris clair).
`depthColors`	Les deux	Définit les couleurs d'arrière-plan pour les lignes en fonction de la profondeur de chaque nœud. La valeur est une palette de couleurs où le premier élément est la couleur d'arrière-plan du nœud racine, le second élément la couleur d'arrière-plan pour ses enfants, etc. en passant en revue le nombre de couleurs fournies dans la palette. Cette propriété de style n'est pas définie par défaut.
`borderStyle`	Les deux	Le composant Tree utilise une occurrence de RectBorder en tant que bordure et répond au styles définis pour cette classe. Voir « Classe RectBorder », à la page 1111. Le style de bordure par défaut est « `inset` ».
`color`	Les deux	Couleur du texte.
`disabledColor`	Les deux	Couleur du texte lorsque le composant est désactivé. La couleur par défaut est 0x848384 (gris foncé).
`embedFonts`	Les deux	Valeur booléenne qui indique si la police spécifiée dans `fontFamily` est de type intégré. Ce style doit être défini sur `true` si `fontFamily` fait référence à une police intégrée. Dans le cas contraire, la police intégrée n'est pas utilisée. Si ce style est défini sur `true` et que `fontFamily` ne fait pas référence à une police intégrée, aucun texte n'est affiché. La valeur par défaut est `false`.

Style	Thème	Description
`fontFamily`	Les deux	Nom de la police du texte. La valeur par défaut est `"_sans"`.
`fontSize`	Les deux	Taille, en points, de la police. La valeur par défaut est 10.
`fontStyle`	Les deux	Style de police : `"normal"` ou `"italic"`. La valeur par défaut est `"normal"`.
`fontWeight`	Les deux	Epaisseur de la police : `"none"` ou `"bold"`. La valeur par défaut est `"none"`. Tous les composants peuvent également accepter la valeur `"normal"` au lieu de `"none"` pendant un appel à la méthode `setStyle()`, mais les appels suivants à `getStyle()` renvoient `"none"`.
`textAlign`	Les deux	Alignement du texte : `"left"`, `"right"` ou `"center"`. La valeur par défaut est `"left"`.
`textDecoration`	Les deux	Décoration du texte : `"none"` ou `"underline"`. La valeur par défaut est `"none"`.
`textIndent`	Les deux	Nombre indiquant le retrait du texte. La valeur par défaut est 0.
`defaultLeafIcon`	Les deux	Icône affichée dans un contrôle Tree pour des nœuds feuilles lorsque aucune icône n'est spécifiée pour un nœud particulier. La valeur par défaut est `"TreeNodeIcon"`, qui est une image représentant un bout de papier.
`disclosureClosedIcon`	Les deux	Icône affichée à côté d'un nœud de dossier fermé dans un composant Tree. La valeur par défaut est `"TreeDisclosureClosed"`, qui est une flèche grise pointant vers la droite.
`disclosureOpenIcon`	Les deux	Icône affichée à côté d'un nœud de dossier ouvert dans un composant Tree. La valeur par défaut est `"TreeDisclosureOpen"`, qui est une flèche grise pointant vers le bas.
`folderClosedIcon`	Les deux	Icône affichée pour un nœud de dossier fermé dans un composant Tree si aucune icône propre à un nœud n'est définie. La valeur par défaut est `"TreeFolderClosed"`, qui est une image de dossier de fichier jaune fermé.

Style	Thème	Description
`folderOpenIcon`	Les deux	Icône affichée pour un nœud de dossier ouvert dans un composant Tree si aucune icône propre à un nœud n'est définie. La valeur par défaut est `"TreeFolderOpen"`, qui est une image de dossier de fichier jaune ouvert.
`indentation`	Les deux	Nombre (en pixels) indiquant le retrait de chaque ligne d'un composant Tree. La valeur par défaut est 17.
`openDuration`	Les deux	Durée, en millisecondes, des animations de développement et de réduction. La valeur par défaut est 250.
`openEasing`	Les deux	Référence à une fonction d'interpolation qui contrôle les animations de développement et de réduction. Par défaut, sine in/out. Pour plus d'informations, consultez « Personnalisation des animations de composants » dans *Utilisation des composants ActionScript 2.0*.
`repeatDelay`	Les deux	Nombre de millisecondes de délai entre le moment où un utilisateur appuie sur un bouton dans la barre de défilement pour la première fois et le moment où l'action commence à se répéter. La valeur par défaut est 500 (une demi-seconde).
`repeatInterval`	Les deux	Nombre de millisecondes entre les clics automatiques lorsqu'un utilisateur maintient le bouton de la souris enfoncé sur un bouton dans la barre de défilement. La valeur par défaut est 35.
`rollOverColor`	Les deux	Couleur d'arrière-plan d'une ligne survolée. La valeur par défaut est 0xE3FFD6 (vert vif) avec le thème Halo et 0xAAAAAA (gris clair) avec le thème Sample. Lorsque `themeColor` est modifié au moyen d'un appel à setStyle(), la structure définit `rollOverColor` sur une valeur liée au `themeColor` choisi.

Style	Thème	Description
selectionColor	Les deux	Couleur d'arrière-plan d'une ligne sélectionnée. La valeur par défaut est 0xCDFFC1 (vert clair) avec le thème Halo et 0xEEEEEE (gris très clair) avec le thème Sample. Lorsque themeColor est modifié au moyen d'un appel à setStyle(), la structure définit selectionColor sur une valeur liée au themeColor choisi.
selectionDuration	Les deux	Longueur (en millisecondes) de la transition d'un état normal à un état sélectionné ou d'un état sélectionné à normal. La valeur par défaut est 200.
selectionDisabledColor	Les deux	Couleur d'arrière-plan d'une ligne sélectionnée. La valeur par défaut est 0xDDDDDD (gris moyen). Etant donné que la valeur par défaut de cette propriété est identique à la valeur par défaut de backgroundDisabledColor, la sélection est invisible lorsque le composant est désactivé à moins que l'une de ces propriétés de style soit modifiée.
selectionEasing	Les deux	Référence à l'équation d'accélération utilisée pour contrôler la transition entre des états de sélection. Ne s'applique qu'à la transition d'un état normal à un état sélectionné. L'équation par défaut utilise une formule sine in/out. Pour plus d'informations, consultez « Personnalisation des animations de composants » dans *Utilisation des composants ActionScript 2.0*.
textRollOverColor	Les deux	Couleur du texte lorsque le pointeur passe dessus. La valeur par défaut est 0x2B333C (gris foncé). Ce style est important lorsque vous définissez rollOverColor, car les deux paramètres doivent se compléter de façon à ce que ce texte soit facilement visualisable pendant un survol.
textSelectedColor	Les deux	Couleur du texte dans la ligne sélectionnée. La valeur par défaut est 0x005F33 (gris foncé). Ce style est important lorsque vous définissez selectionColor, car les deux paramètres doivent se compléter pour que ce texte soit facilement visualisable lors de la sélection.
useRollOver	Les deux	Détermine si le survol d'une ligne active sa mise en surbrillance. La valeur par défaut est true.

Par exemple, le code suivant crée une occurrence Tree `first_tr` à l'aide de `UIObject.createClassObject()`, remplit l'arborescence en utilisant un fournisseur de données, puis utilise `UIObject.setStyle()` pour modifier l'indentation des nœuds de l'arborescence sur 8 pixels. Faites glisser un composant Tree vers la bibliothèque du document en cours, puis ajoutez le code suivant à la première image du scénario principal :

```
import mx.controls.Tree;

this.createClassObject(Tree, "first_tr", 20);
first_tr.setSize(200, 100);
first_tr.move(0, 120);

var trDP_xml:XML = new XML("<node label='1st Local Folder'><node
  label='Inbox' data='0'/><node label='Outbox' data='1'/><node label='2nd
  Local Folder'><node label='Inbox' data='0'/><node label='Outbox'
  data='1'/></node></node>");
first_tr.dataProvider = trDP_xml;

first_tr.setStyle("indentation", 8);
```

Définition des styles pour tous les composants Tree dans un document

La classe Tree hérite de la classe List qui hérite de la classe ScrollSelectList. Les propriétés de style de niveau de classe par défaut sont définies sur la classe ScrollSelectList que le composant Menu et tous les composants basés sur des listes étendent. Vous pouvez définir de nouvelles valeurs de style par défaut directement sur cette classe, les nouveaux paramètres sont alors reflétés dans tous les composants concernés.

```
_global.styles.ScrollSelectList.setStyle("backgroundColor", 0xFF00AA);
```

Pour définir une propriété de style sur les composants Tree uniquement, vous pouvez créer une occurrence `CSSStyleDeclaration` et l'enregistrer dans `_global.styles.DataGrid`.

```
import mx.styles.CSSStyleDeclaration;
if (_global.styles.Tree == undefined) {
  _global.styles.Tree = new CSSStyleDeclaration();
}
_global.styles.Tree.setStyle("backgroundColor", 0xFF00AA);
```

Lors de la création d'une déclaration de style de niveau classe, toutes les valeurs par défaut fournies par la déclaration `ScrollSelectList` sont perdues. Ceci inclut `backgroundColor` qui est nécessaire pour prendre en charge les événements de souris. Pour créer une déclaration de style de niveau de classe et conserver les valeurs par défaut, utilisez une boucle `for`, comme suit, pour copier les anciens paramètres dans la nouvelle déclaration.

```
var source = _global.styles.ScrollSelectList;
var target = _global.styles.Tree;
for (var style in source) {
  target.setStyle(style, source.getStyle(style));
}
```

Utilisation d'enveloppes avec le composant Tree

Le composant Tree utilise une occurrence de RectBorder pour sa bordure et des barres de défilement pour des images de défilement. Pour plus d'informations sur l'application d'enveloppe à ces éléments visuels, reportez-vous à « Classe RectBorder », à la page 1111 et à « Utilisation d'enveloppes avec le composant UIScrollBar », à la page 1448.

Classe Tree

Héritage MovieClip > Classe UIObject > Classe UIComponent > View > ScrollView > ScrollSelectList > Composant List > Tree

Nom de classe Actionscript mx.controls.Tree

Les méthodes, propriétés et événements de la classe Tree vous permettent de gérer et de manipuler des objets Tree.

Méthodes de la classe Tree

Le tableau suivant répertorie les méthodes de la classe Tree.

Méthode	Description
`Tree.addTreeNode()`	Ajoute un nœud dans une occurrence de Tree.
`Tree.addTreeNodeAt()`	Ajoute un nœud à un endroit spécifique d'une occurrence de Tree.
`Tree.getDisplayIndex()`	Renvoie l'index d'affichage d'un nœud donné.
`Tree.getIsBranch()`	Précise si le dossier est une branche (possède une icône de dossier et une flèche d'agrandissement).
`Tree.getIsOpen()`	Indique si un nœud est ouvert ou fermé.

Méthode	Description
`Tree.getNodeDisplayedAt()`	Etablit la correspondance entre un index d'affichage de l'arborescence et un nœud affiché.
`Tree.getTreeNodeAt()`	Renvoie un nœud à la racine de l'arborescence.
`Tree.refresh()`	Met à jour l'arborescence.
`Tree.removeAll()`	Supprime tous les nœuds d'une occurrence de Tree et actualise l'arborescence.
`Tree.removeTreeNodeAt()`	Supprime un nœud à un endroit indiqué et actualise l'arborescence.
`Tree.setIcon()`	Définit une icône pour le nœud spécifié.
`Tree.setIsBranch()`	Indique si un nœud est une branche (comporte une icône de dossier et une flèche d'agrandissement).
`Tree.setIsOpen()`	Ouvre ou ferme un nœud.

Méthodes héritées de la classe UIObject

Le tableau suivant répertorie les méthodes de la classe Tree héritées de la classe UIObject. Pour appeler ces méthodes à partir de l'objet Tree, utilisez le formulaire *`TreeInstance.methodName`*.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
`UIObject.setSize()`	Redimensionne l'objet à la taille demandée.
`UIObject.setSkin()`	Définit une enveloppe dans l'objet.
`UIObject.setStyle()`	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant répertorie les méthodes de la classe Tree héritées de la classe UIComponent. Pour appeler ces méthodes à partir de l'objet Tree, utilisez le formulaire *`TreeInstance.methodName`*.

Méthode	Description
`UIComponent.getFocus()`	Renvoie une référence à l'objet ayant le focus.
`UIComponent.setFocus()`	Attribue le focus à l'occurrence de composant.

Méthodes héritées de la classe List

Le tableau suivant répertorie les méthodes de la classe Tree héritées de la classe List. Pour appeler ces méthodes à partir de l'objet Tree, utilisez le formulaire *`TreeInstance.methodName`*.

Méthode	Description
`List.addItem()`	Ajoute un élément à la fin de la liste.
`List.addItemAt()`	Ajoute un élément à la liste, à l'index spécifié. Il est préférable d'utiliser `Tree.addTreeNodeAt()` avec le composant Tree.
`List.getItemAt()`	Renvoie l'élément à l'emplacement d'index spécifié.
`List.removeAll()`	Supprime tous les éléments de la liste.
`List.removeItemAt()`	Supprime l'élément à l'index spécifié.
`List.replaceItemAt()`	Remplace l'élément, à l'index spécifié, par un autre élément.
`List.setPropertiesAt()`	Applique les propriétés spécifiées à un élément donné.
`List.sortItems()`	Trie les éléments de la liste à l'aide de la fonction de comparaison spécifiée.
`List.sortItemsBy()`	Trie les éléments de la liste à l'aide d'une propriété donnée.

Récapitulatif des propriétés de la classe Tree

Le tableau suivant répertorie les propriétés de la classe Tree.

Propriété	Description
`Tree.dataProvider`	Indique une source de données XML.
`Tree.firstVisibleNode`	Indique le premier nœud en haut de l'affichage.
`Tree.selectedNode`	Indique le nœud sélectionné dans une occurrence de Tree.
`Tree.selectedNodes`	Indique les nœuds sélectionnés dans une occurrence de Tree.

Propriétés héritées de la classe UIObject

Le tableau suivant énumère les propriétés de la classe Tree héritées de la classe UIObject. Pour accéder à ces propriétés à partir de l'objet Tree, utilisez le formulaire *`TreeInstance.propertyName`*.

Propriété	Description
`UIObject.bottom`	Lecture seule ; position du bord inférieur de l'objet par rapport au bord inférieur de son parent.
`UIObject.height`	Lecture seule ; hauteur de l'objet, en pixels.
`UIObject.left`	Lecture seule ; bord gauche de l'objet, en pixels.
`UIObject.right`	Lecture seule ; position du bord droit de l'objet par rapport au bord droit de son parent.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Lecture seule ; renvoie la position du bord supérieur de l'objet par rapport à son parent.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Lecture seule ; largeur de l'objet, en pixels.
`UIObject.x`	Lecture seule ; bord gauche de l'objet, en pixels.
`UIObject.y`	Lecture seule ; bord supérieur de l'objet, en pixels.

Propriétés héritées de la classe UIComponent

Le tableau suivant répertorie les propriétés de la classe Tree héritées de la classe UIComponent. Pour accéder à ces propriétés à partir de l'objet Tree, utilisez le formulaire *TreeInstance.propertyName*.

Propriété	Description
UIComponent.enabled	Indique si le composant peut recevoir le focus et la saisie.
UIComponent.tabIndex	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Propriétés héritées de la classe List

Le tableau suivant répertorie les propriétés de la classe Tree héritées de la classe List. Pour accéder à ces propriétés à partir de l'objet Tree, utilisez le formulaire *TreeInstance.propertyName*.

Propriété	Description
List.cellRenderer	Affecte la classe ou le symbole à utiliser pour afficher chaque ligne de la liste.
List.dataProvider	Source des éléments de la liste.
List.hPosition	Position horizontale de la liste.
List.hScrollPolicy	Indique si la barre de défilement horizontale est affichée (« on ») ou non (« off »).
List.iconField	Champ situé dans chaque élément et utilisé pour spécifier les icônes.
List.iconFunction	Fonction qui détermine les icônes à utiliser.
List.labelField	Spécifie un champ dans chaque élément, à utiliser comme texte d'étiquette.
List.labelFunction	Fonction qui détermine les champs de chaque élément à utiliser pour le texte d'étiquette.
List.length	Nombre d'éléments de la liste. Cette propriété est en lecture seule.
List.maxHPosition	Nombre de pixels que la liste peut faire défiler à droite, lorsque List.hScrollPolicy est défini sur "on".
List.multipleSelection	Indique si la sélection multiple est autorisée dans la liste (true) ou non (false).

Propriété	Description
`List.rowCount`	Nombre de lignes au moins partiellement visibles dans la liste.
`List.rowHeight`	Hauteur des pixels de chaque ligne de la liste.
`List.selectable`	Indique si la liste peut être sélectionnée (`true`) ou non (`false`).
`List.selectedIndex`	Index d'une sélection dans une liste à sélection unique.
`List.selectedIndices`	Tableau des éléments sélectionnés dans une liste à sélection multiple.
`List.selectedItem`	Elément sélectionné dans une liste à sélection unique. Cette propriété est en lecture seule.
`List.selectedItems`	Objets sélectionnés dans une liste à sélection multiple. Cette propriété est en lecture seule.
`List.vPosition`	Fait défiler la liste pour que le premier élément visible soit le numéro affecté.
`List.vScrollPolicy`	Indique si la barre de défilement verticale est affichée (« `on` »), ne l'est pas (« `off` ») ou affichée si nécessaire (« `auto` »).

Récapitulatif des événements de la classe Tree

Le tableau suivant répertorie les événements de la classe Tree.

Evénement	Description
`Tree.nodeClose`	Diffusé lorsqu'un nœud est fermé par un utilisateur.
`Tree.nodeOpen`	Diffusé lorsqu'un nœud est ouvert par un utilisateur.

Evénements hérités de la classe UIObject

Le tableau suivant répertorie les événements de la classe Tree hérités de la classe UIObject.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant répertorie les événements de la classe Tree hérités de la classe UIComponent.

Evénement	Description
`UIComponent.focusIn`	Diffusé lorsqu'un objet reçoit le focus.
`UIComponent.focusOut`	Diffusé lorsqu'un objet perd le focus.
`UIComponent.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIComponent.keyUp`	Diffusé lorsqu'une touche est relâchée.

Evénements hérités de la classe List

Le tableau suivant répertorie les événements de la classe Tree hérités de la classe List.

Evénement	Description
`List.change`	Diffusé chaque fois que la sélection change suite à une interaction avec l'utilisateur.
`List.itemRollOut`	Diffusé lorsque le pointeur survole, puis quitte un élément de la liste.
`List.itemRollOver`	Diffusé lorsque le pointeur survole des éléments de la liste.
`List.scroll`	Diffusé lors du défilement d'une liste.

Tree.addTreeNode()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
treeInstance.addTreeNode(label [, data])
```

Utilisation 2 :

```
treeInstance.addTreeNode(child)
```

Paramètres

label Chaîne qui affiche le nœud, ou un objet possédant un champ label (ou n'importe quel nom de champ d'étiquette spécifié par la propriété `labelField`).

data Objet de n'importe quel type associé au nœud. Ce paramètre est facultatif.

child Tout objet XMLNode.

Renvoie

Le nœud XML ajouté.

Description

Méthode : ajoute un nœud enfant à l'arborescence. Le nœud est construit à partir des informations fournies dans les paramètres *label* et *data* (Utilisation 1) ou à partir du nœud enfant (objet XMLNode) prédéfini (Utilisation 2). Si le nœud ajouté existe déjà, il est supprimé de son emplacement précédent.

L'appel de cette méthode actualise l'affichage.

Exemple

L'exemple suivant crée deux composants Tree avec un nœud chacun, respectivement `1st Local Folders` et `2nd Local Folders`. Il ajoute ensuite le nœud de l'arborescence (`2nd Local Folders`) du deuxième composant Tree au premier composant Tree et ajoute également un nouveau nœud, `Inbox`.

Vous devez d'abord ajouter le composant à la bibliothèque du document en faisant glisser un composant Tree sur la scène et en le supprimant ensuite ; ajoutez ensuite le code suivant à l'image 1.

CONSEIL	Essayez d'abord cet exemple sans les deux instructions `addTreeNode()` à la fin, puis essayez l'exemple complet.

```
/**
 Requiert :
  - Composant Tree dans la bibliothèque
*/

import mx.controls.Tree;

this.createClassObject(Tree, "first_tr", 10);
first_tr.setSize(200, 100);

this.createClassObject(Tree, "second_tr", 20);
second_tr.setSize(200, 100);
second_tr.move(0, 120);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0'/><node label='Outbox' data='1'/></node>");
first_tr.dataProvider = trDP_xml;

var trDP2_xml:XML = new XML("<node label='2nd Local Folders'><node
  label='Outbox' data='0'/><node label='Outbox' data='1'/></node>");
second_tr.dataProvider = trDP2_xml;

// Ajout du nœud de second_tr à first_tr.
first_tr.addTreeNode(second_tr.getTreeNodeAt(0));

// Ajout du nœud à first_tr.
first_tr.addTreeNode("Inbox", "data");
```

Tree.addTreeNodeAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

Utilisation 1 :

```
treeInstance.addTreeNodeAt(index, label [, data])
```

Utilisation 2 :

```
treeInstance.addTreeNodeAt(index, child)
```

Paramètres

index Position d'index basée sur zéro (parmi les nœuds enfants) à laquelle le nœud doit être ajouté.

label Chaîne qui contient le nom du nœud à ajouter.

data Objet de n'importe quel type associé au nœud. Ce paramètre est facultatif.

child Tout objet XMLNode.

Valeur renvoyée

Le nœud XML ajouté.

Description

Méthode : ajoute un nœud à l'endroit spécifié de l'arborescence. Le nœud est construit à partir des informations fournies dans les paramètres *label* et *data*(Utilisation 1) ou à partir de l'objet XMLNode prédéfini (Utilisation 2). Si le nœud ajouté existe déjà, il est supprimé de son emplacement précédent.

L'appel de cette méthode actualise l'affichage.

Exemple

L'exemple suivant crée deux composants Tree avec un nœud pour chacun d'eux : `1st Local Folders` et `2nd Local Folders`, respectivement. Il fait appel à la première utilisation de `addTreeNodeAt()` pour ajouter un nouveau nœud, `Inbox`, au premier composant Tree. Il fait appel ensuite à la seconde utilisation de `addTreeNodeAt()` pour ajouter le 1er nœud (`getTreeNodeAt(0)`) du deuxième composant Tree au premier composant Tree.

Vous devez d'abord ajouter le composant à la bibliothèque du document en faisant glisser un composant Tree sur la scène et en le supprimant ensuite ; ajoutez ensuite le code suivant à l'image 1.

CONSEIL	Essayez d'abord cet exemple sans les deux instructions `addTreeNodeAt()` à la fin, puis essayez l'exemple complet.

```
/**
 Requiert :
  - Composant Tree dans la bibliothèque
*/

import mx.controls.Tree;

this.createClassObject(Tree, "first_tr", 10);
first_tr.setSize(200, 100);

this.createClassObject(Tree, "second_tr", 20);
second_tr.setSize(200, 100);
second_tr.move(0, 120);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0'/><node label='Outbox' data='1'/></node>");
first_tr.dataProvider = trDP_xml;

var trDP2_xml:XML = new XML("<node label='2nd Local Folders'><node
  label='Outbox' data='0'/><node label='Outbox' data='1'/></node>");
second_tr.dataProvider = trDP2_xml;

// Ajout du nœud à first_tr.
first_tr.addTreeNodeAt(1, "Inbox", "data");
// Ajout du nœud de second_tr à first_tr.
first_tr.addTreeNodeAt(2, second_tr.getTreeNodeAt(0));
```

Tree.dataProvider

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
treeInstance.dataProvider
```

Description

Propriété : XML ou une chaîne. Si `dataProvider` est un objet XML, il est ajouté directement dans l'arborescence. Si `dataProvider` est une chaîne, il doit contenir un objet XML valide lu par l'arborescence et converti en objet XML.

Vous pouvez charger l'objet XML à partir d'une source externe lors de l'exécution ou le créer dans Flash pendant la programmation. Pour créer un objet XML, vous avez la possibilité d'utiliser les méthodes TreeDataProvider ou les méthodes et propriétés de la classe XML intégrée d'ActionScript. Vous pouvez également créer une chaîne contenant un objet XML.

Les objets XML qui se trouvent sur la même image sous forme de composant Tree contiennent automatiquement les méthodes et propriétés TreeDataProvider. Vous pouvez utiliser les objets XML ou XMLNode d'ActionScript.

Exemple

L'exemple suivant utilise la propriété `dataProvider` pour ajouter le contenu d'un fichier XML à l'occurrence `my_tr` du composant Tree :

Vous devez d'abord ajouter une occurrence du composant Tree sur la scène et la nommer `my_tr` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Composant Tree sur la scène (nom d'occurrence : my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML();
trDP_xml.ignoreWhite = true;
trDP_xml.onLoad = function(success:Boolean){
   my_tr.dataProvider = trDP_xml.firstChild;
}
trDP_xml.load("http://www.flash-mx.com/mm/xml/tree.xml");
```

REMARQUE

La plupart des fichiers XML contiennent des espaces. Pour que Flash ignore les espaces vierges, vous devez définir la propriété `XML.ignoreWhite` sur `true`.

Tree.firstVisibleNode

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
treeInstance.firstVisibleNode = someNode
```

Description

Propriété : indique le premier nœud visible en haut de l'affichage de l'arborescence. Utilisez cette propriété pour faire défiler l'affichage de l'arborescence sur une position donnée. Si le nœud spécifié *someNode* se trouve dans un nœud qui n'est pas développé, configurer `firstVisibleNode` n'a aucun effet. La valeur par défaut correspond au premier nœud visible ou est `undefined` s'il n'existe aucun nœud visible. La valeur de cette propriété est un objet XMLNode.

Cette propriété est semblable à la propriété `List.vPosition`.

Exemple

L'exemple suivant remplit un composant Tree appelé `my_tr` avec six nœuds qu'il crée depuis une chaîne de texte XML. Il rend le dernier nœud visible dans le composant Tree en affectant le dernier nœud (position relative 5) à la propriété `firstVisibleNode`. Vous pouvez rendre d'autres nœuds visibles en modifiant la valeur 5 sur d'autres valeurs comprises entre 0 et 4.

Vous devez d'abord ajouter une occurrence du composant Tree sur la scène et la nommer `my_tr` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Composant Tree sur la scène (nom d'occurrence : my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);
```

```
var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0' /><node label='Outbox' data='1' /></node><node
  label='2nd Local Folders'><node label='Inbox' data='0' /><node
  label='Outbox' data='1' /></node><node label='3rd Local Folders'><node
  label='Inbox' data='0' /><node label='Outbox' data='1' /></node><node
  label='4th Local Folders'><node label='Inbox' data='0' /><node
  label='Outbox' data='1' /></node><node label='5th Local Folders'><node
  label='Inbox' data='0' /><node label='Outbox' data='1' /></node><node
  label='6th Local Folders'><node label='Inbox' data='0' /><node
  label='Outbox' data='1' /></node>");
my_tr.dataProvider = trDP_xml;

// Définition du dernier nœud comme nœud visible.
my_tr.firstVisibleNode = my_tr.getTreeNodeAt(5);
```

Tree.getDisplayIndex()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`treeInstance.getDisplayIndex(`*`node`*`)`

Paramètres

`node` Objet XMLNode.

Valeur renvoyée

Index du nœud spécifié ou `undefined` si le nœud n'est pas actuellement affiché.

Description

Méthode : renvoie l'index d'affichage du nœud spécifié dans le paramètre *`node`*.

L'index d'affichage indique la position de l'élément dans la liste des éléments visibles dans la fenêtre de l'arborescence. Par exemple, les enfants d'un nœud fermé ne se trouvent pas dans l'index d'affichage. La liste des index d'affichage commence à 0 et répertorie les éléments visibles, quels que soient leurs parents. En d'autres termes, l'index correspond au numéro (commençant par 0) des lignes affichées.

Exemple

L'exemple suivant ajoute six nœuds à un composant Tree et appelle la méthode `getDisplayIndex()` pour afficher la position du nœud sélectionné par l'utilisateur.

Vous devez d'abord ajouter une occurrence du composant Tree sur la scène et la nommer `my_tr` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Composant Tree sur la scène (nom d'occurrence : my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 140);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0' /><node label='Outbox' data='1' /></node><node
  label='2nd Local Folders'><node label='Inbox' data='0' /><node
  label='Outbox' data='1' /></node><node label='3rd Local Folders'><node
  label='Inbox' data='0' /><node label='Outbox' data='1' /></node><node
  label='4th Local Folders'><node label='Inbox' data='0' /><node
  label='Outbox' data='1' /></node><node label='5th Local Folders'><node
  label='Inbox' data='0' /><node label='Outbox' data='1' /></node><node
  label='6th Local Folders'><node label='Inbox' data='0' /><node
  label='Outbox' data='1' /></node>");
my_tr.dataProvider = trDP_xml;
my_tr.firstVisibleNode = my_tr.getTreeNodeAt(0);

my_tr.addEventListener("change", listChanged);
function listChanged(evt_obj:Object) {
  trace(my_tr.getDisplayIndex(evt_obj.target.selectedNode));
}
```

Tree.getIsBranch()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
treeInstance.getIsBranch(node)
```

Paramètres

node Objet XMLNode.

Valeur renvoyée

Une valeur Booléenne qui indique si le nœud est une branche (`true`) ou non (`false`).

Description

Méthode : indique si le nœud spécifié possède une icône de dossier et une flèche d'agrandissement (et est par conséquent une branche). Cette configuration est automatique lorsque des enfants sont ajoutés au nœud. Il vous suffit simplement d'appeler la méthode `Tree.setIsBranch()` pour créer des dossiers vides.

Exemple

L'exemple suivant crée deux nœuds dans une arborescence et appelle `isBranch()` pour déterminer si le deuxième est une branche.

Vous devez d'abord ajouter une occurrence du composant Tree sur la scène et la nommer `my_tr` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Composant Tree sur la scène (nom d'occurrence : my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0' /><node label='Outbox' data='1' /></node><node
  label='2nd Local Folders'><node label='Inbox' data='2' /><node
  label='Outbox' data='3' /></node>");
my_tr.dataProvider = trDP_xml;

var isBranch:Boolean = my_tr.getIsBranch(my_tr.getTreeNodeAt(1));
trace("2nd node is a branch: " + isBranch);
```

Voir aussi

`Tree.setIsBranch()`

Tree.getIsOpen()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
treeInstance.getIsOpen(node)
```

Paramètres

node Objet XMLNode.

Valeur renvoyée

Une valeur Booléenne indiquant si l'arborescence est ouverte (`true`) ou fermée (`false`).

Description

Méthode : indique si le nœud spécifié est ouvert ou fermé.

REMARQUE	Les index d'affichage changent à chaque ouverture ou fermeture de nœud.

Exemple

L'exemple suivant ajoute deux nœuds à un composant Tree appelé `my_tr`, ouvre le deuxième nœud, puis appelle `getIsOpen()` pour afficher l'état du deuxième nœud.

Vous devez d'abord ajouter une occurrence du composant Tree sur la scène et la nommer `my_tr` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Composant Tree sur la scène (nom d'occurrence : my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);
```

```
var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0'/><node label='Outbox' data='1'/></node><node
  label='2nd Local Folders'><node label='Inbox' data='2'/><node
  label='Outbox' data='3'/></node>");
my_tr.dataProvider = trDP_xml;
my_tr.setIsOpen(my_tr.getTreeNodeAt(1), true);
var isOpen:Boolean = my_tr.getIsOpen(my_tr.getTreeNodeAt(1));
trace("2nd node is a open: " + isOpen);
```

Tree.getNodeDisplayedAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`treeInstance.getNodeDisplayedAt(`*`index`*`)`

Paramètres

`index` Nombre entier représentant la position d'affichage dans la zone de l'arborescence qui peut être affichée. Ce nombre est basé sur zéro ; le nœud en première position est 0, celui en deuxième position est 1 et ainsi de suite.

Valeur renvoyée

L'objet XMLNode spécifié.

Description

Méthode : permet d'établir la correspondance entre un index d'affichage de l'arborescence et un nœud affiché. Par exemple, si la cinquième ligne de l'arborescence a affiché un nœud se trouvant à huit niveaux de profondeur dans la hiérarchie, ce nœud est renvoyé par un appel à `getNodeDisplayedAt(4)`.

L'index d'affichage est un tableau d'éléments pouvant s'afficher dans la fenêtre de l'arborescence. Par exemple, les enfants d'un nœud fermé ne se trouvent pas dans l'index d'affichage. L'index d'affichage commence à 0 et répertorie les éléments visibles, quels que soient leurs parents. En d'autres termes, l'index d'affichage correspond au numéro (commençant par 0) des lignes affichées.

REMARQUE	Les index d'affichage changent à chaque ouverture ou fermeture de nœud.

Exemple

L'exemple suivant ajoute un nœud à une occurrence Tree appelée `my_tr`, puis appelle la méthode `getNodeDisplayedAt()` pour récupérer le nœud à la position d'affichage 0 (zéro). L'exemple appelle la fonction trace() pour afficher le nœud.

Vous devez d'abord ajouter une occurrence du composant Tree sur la scène et la nommer `my_tr` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Composant Tree sur la scène (nom d'occurrence : my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label=\"1st Local Folders\"><node
  label=\"Inbox\" data=\"0\" />");
my_tr.dataProvider = trDP_xml;

trace(my_tr.getNodeDisplayedAt(0));
```

Tree.getTreeNodeAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
treeInstance.getTreeNodeAt(index)
```

Paramètres

index Numéro d'index d'un nœud.

Valeur renvoyée

Un Objet XMLNode.

Description

Méthode : renvoie le nœud spécifié sur la racine de `myTree`.

Exemple

L'exemple suivant ajoute un nœud à l'occurrence Tree appelée `my_tr`, puis appelle la méthode `getTreeNodeAt()` pour renvoyer le premier nœud dans l'arborescence.

Vous devez d'abord ajouter une occurrence du composant Tree sur la scène et la nommer `my_tr` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Composant Tree sur la scène (nom d'occurrence : my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label=\"1st Local Folders\"><node
  label=\"Inbox\" data=\"0\" /><node label=\"Outbox\" data=\"1\" /></
  node>");
my_tr.dataProvider = trDP_xml;

trace(my_tr.getTreeNodeAt(0));
```

Tree.nodeClose

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.nodeClose = function(eventObject:Object) {
  // Insertion du code ici.
};
treeInstance.addEventListener("nodeClose", listenerObject);
```

Description

Evénement : diffusé vers l'ensemble des écouteurs enregistrés lorsque les nœuds d'un composant Tree sont fermés par un utilisateur.

Les composants utilisent un modèle d'événement dispatcher(diffuseur)/écouteur.
Le composant Tree diffuse un événement `nodeClose` lorsque l'utilisateur clique sur l'un de ses nœuds pour le fermer et lorsque l'événement est géré par une fonction, aussi appelée *gestionnaire*, associée à un objet écouteur (`listenerObject`) que vous créez.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (`eventObject`) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. L'objet événement de l'événement `Tree.nodeClose` possède une propriété supplémentaire : `node` (nœud XML fermé).

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

L'exemple suivant ajoute deux nœuds à l'occurrence Tree `my_tr`, puis crée deux objets écouteur, un pour les événements `nodeOpen` et un autre pour les événements `nodeClose`. Lorsque ces événements ont lieu, la fonction écouteur utilise une instruction trace pour afficher l'événement et le nœud affecté dans le panneau Sortie.

Vous devez d'abord ajouter une occurrence du composant Tree sur la scène et la nommer `my_tr` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Composant Tree sur la scène (nom d'occurrence : my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0'/><node label='Outbox' data='1'/></node><node
  label='2nd Local Folders'><node label='Inbox' data='2'/><node
  label='Outbox' data='3'/></node>");
my_tr.dataProvider = trDP_xml;

// Création d'un objet écouteur.
var trListener:Object = new Object();
trListener.nodeOpen = function(evt_obj:Object){
   trace("Node opened\n" + evt_obj.node);
   trace("\n");
}
```

```
trListener.nodeClose = function(evt_obj:Object){
   trace("Node closed\n" + evt_obj.node);
   trace("\n");
}
// Ajout d'écouteurs.
my_tr.addEventListener("nodeOpen", trListener);
my_tr.addEventListener("nodeClose", trListener);
```

Tree.nodeOpen

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
var listenerObject:Object = new Object();
listenerObject.nodeOpen = function(eventObject:Object) {
  // Insertion du code ici.
};
treeInstance.addEventListener("nodeOpen", listenerObject);
```

Description

Evénement : diffusé vers l'ensemble des objets d'écoute enregistrés lorsqu'un utilisateur ouvre un nœud sur un composant Tree.

Les composants utilisent un modèle d'événement dispatcher(diffuseur)/écouteur.
Le composant Tree distribue un événement `nodeOpen` lorsque l'utilisateur clique sur un nœud pour l'ouvrir et lorsque l'événement est géré par une fonction, appelée aussi *gestionnaire*, associée à un objet écouteur (*`listenerObject`*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`eventObject`*) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. L'objet événement de l'événement `Tree.nodeOpen` possède une propriété supplémentaire : `node` (le nœud XML ouvert).

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

L'exemple suivant ajoute deux nœuds à l'occurrence Tree `my_tr`, puis crée deux objets écouteur, un pour les événements `nodeOpen` et un autre pour les événements `nodeClose`. Lorsque ces événements ont lieu, les fonctions écouteur appellent des instructions trace pour afficher l'événement et le nœud affecté dans le panneau Sortie.

Vous devez d'abord ajouter une occurrence du composant Tree sur la scène et la nommer `my_tr` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Composant Tree sur la scène (nom d'occurrence : my_tr)
*/
var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0'/><node label='Outbox' data='1'/></node><node
  label='2nd Local Folders'><node label='Inbox' data='2'/><node
  label='Outbox' data='3'/></node>");
my_tr.dataProvider = trDP_xml;

// Création d'un objet écouteur.
var trListener:Object = new Object();
trListener.nodeOpen = function(evt_obj:Object){
   trace("Node opened\n" + evt_obj.node);
   trace("\n");
}
trListener.nodeClose = function(evt_obj:Object){
   trace("Node closed\n" + evt_obj.node);
   trace("\n");
}
// Ajout d'écouteurs.
my_tr.addEventListener("nodeOpen", trListener);
my_tr.addEventListener("nodeClose", trListener);
```

Tree.refresh()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
treeInstance.refresh()
```

Paramètres

Aucun.

Valeur renvoyée

Aucune.

Description

Méthode : met à jour l'arborescence.

Exemple

L'exemple suivant ajoute un nœud à une occurrence de composant Tree appelée my_tr et crée des écouteurs pour un bouton Refresh (Actualiser) et un bouton Supprimer tout.
En supposant que la source XML du fournisseur de données ait changé, l'utilisateur peut cliquer sur le bouton Actualiser, puis le code appelle la méthode `refresh()` pour mettre à jour le contenu de l'arborescence.

Vous devez d'abord ajouter une occurrence du composant Tree sur la scène et la nommer `my_tr`. Ajoutez ensuite un bouton nommé `refresh_button`. Ajoutez ensuite le code suivant à l'image 1 du scénario principal :

```
/**
 Requiert :
  - Composant Tree sur la scène (nom d'occurrence : my_tr)
  - Composant Button sur la scène (nom d'occurrence : refresh_button)
*/

var my_tr:mx.controls.Tree;
var refresh_button:mx.controls.Button;
var removeAll_button:mx.controls.Button;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML();
```

```
trDP_xml.ignoreWhite = true;
trDP_xml.onLoad = function() {
 my_tr.dataProvider = this.firstChild;
};
trDP_xml.load("http://yourXMLsourcehere");

function refreshListener(evt_obj:Object):Void {
 my_tr.refresh();
}
refresh_button.addEventListener("click", refreshListener);
```

Tree.removeAll()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
treeInstance.removeAll()
```

Paramètres

Aucun.

Valeur renvoyée

Aucune.

Description

Méthode : supprime tous les nœuds et actualise l'arborescence.

Exemple

L'exemple suivant ajoute un nœud à une occurrence de composant Tree appelée my_tr et crée un écouteur pour un bouton Supprimer tout. Lorsque l'utilisateur clique sur le bouton Supprimer tout, le code appelle la méthode `removeAll()` pour supprimer tous les nœuds de l'arborescence.

Vous ajoutez d'abord une occurrence du composant Tree sur la scène et la nommer `my_tr` ; ajoutez ensuite un bouton appelé `removeAll_button`, puis ajoutez le code suivant à l'image 1.

```
/**
 Requiert :
  - Composant Tree sur la scène (nom d'occurrence : my_tr)
  - Composant Button sur la scène (nom d'occurrence : refresh_button)
  - Composant Button sur la scène (nom d'occurrence : removeAll_button)
*/

var my_tr:mx.controls.Tree;
var removeAll_button:mx.controls.Button;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML();
trDP_xml.ignoreWhite = true;
trDP_xml.onLoad = function() {
 my_tr.dataProvider = this.firstChild;
};
trDP_xml.load("http://www.flash-mx.com/mm/xml/tree.xml");

function removeAllListener(evt_obj:Object):Void {
 my_tr.removeAll();
}
removeAll_button.addEventListener("click", removeAllListener);
```

Tree.removeTreeNodeAt()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`treeInstance.removeTreeNodeAt(`*`index`*`)`

Paramètres

`index` Numéro d'index d'un enfant de l'arborescence. Chaque enfant d'une arborescence est associé à un numéro d'index en fonction de son ordre de création ; le premier numéro d'index est 0.

Valeur renvoyée

Un objet XMLNode ou `undefined` s'il existe une erreur.

Description

Méthode : supprime un nœud (spécifié par sa position dans l'index) à la racine de l'arborescence et actualise cette dernière.

Exemple

L'exemple suivant ajoute deux nœuds à une occurrence de composant Tree et crée un écouteur pour un événement change sur l'arborescence. Lorsqu'un événement change a lieu, les fonctions écouteur appellent la méthode `removeTreeNodeAt()` pour supprimer le nœud sélectionné de l'arborescence.

Vous ajoutez d'abord une occurrence du composant Tree sur la scène et la nommer `my_tr` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Composant Tree sur la scène (nom d'occurrence : my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0' /><node label='Outbox' data='1' /></node><node
  label='2nd Local Folders'><node label='Inbox' data='2' /><node
  label='Outbox' data='3' /></node>");
my_tr.dataProvider = trDP_xml;

var treeListener:Object = new Object();
treeListener.change = function (evt_obj:Object) {
 my_tr.removeTreeNodeAt(my_tr.selectedIndex);
}
my_tr.addEventListener("change", treeListener);
```

Tree.selectedNode

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
treeInstance.selectedNode
```

Description

Propriété : spécifie le nœud sélectionné dans une occurrence d'arborescence.

Exemple

L'exemple suivant ajoute deux nœuds à une occurrence de composant Tree et définit le deuxième nœud sur l'état sélectionné.

Vous devez d'abord ajouter une occurrence du composant Tree sur la scène et la nommer `my_tr` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Composant Tree sur la scène (nom d'occurrence : my_tr)
*/
var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0' /><node label='Outbox' data='1' /></node><node
  label='2nd Local Folders'><node label='Inbox' data='2' /><node
  label='Outbox' data='3' /></node>");
my_tr.dataProvider = trDP_xml;

// Sélection du deuxième nœud.
my_tr.selectedNode = my_tr.getTreeNodeAt(1);
```

Voir aussi

`Tree.selectedNodes`

Tree.selectedNodes

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
treeInstance.selectedNodes
```

Description

Propriété : spécifie les nœuds sélectionnés dans une occurrence d'arborescence.

Exemple

L'exemple suivant ajoute trois nœuds à une occurrence de composant Tree et définit les deux premiers sur l'état sélectionné.

Vous devez d'abord ajouter une occurrence du composant Tree sur la scène et la nommer `my_tr` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Composant Tree sur la scène (nom d'occurrence : my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0'/><node label='Outbox' data='1'/></node><node
  label='2nd Local Folders'><node label='Inbox' data='2'/><node
  label='Outbox' data='3'/></node><node label='3rd Local Folders'><node
  label='Inbox' data='2'/><node label='Outbox' data='3'/></node>");
my_tr.dataProvider = trDP_xml;

// Plusieurs sélections autorisées.
my_tr.multipleSelection = true;

// Sélection du premier et du deuxième nœud.
my_tr.selectedNodes = [my_tr.getTreeNodeAt(0), my_tr.getTreeNodeAt(1)];
```

Voir aussi

`Tree.selectedNode`

Tree.setIcon()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
treeInstance.setIcon(node, linkID [, linkID2])
```

Paramètres

node Nœud XML.

linkID Identifiant de liaison d'un symbole à utiliser en tant qu'icône à côté du nœud. Ce paramètre est utilisé pour les nœuds de feuille et pour l'état fermé des nœuds de branche.

linkID2 Pour un nœud de branche, identifiant de liaison d'un symbole à utiliser en tant qu'icône qui représente l'état ouvert du nœud. Ce paramètre est facultatif.

Renvoie

Aucune.

Description

Méthode : définit une icône pour le nœud spécifié. Cette méthode prend un paramètre ID (*linkID*) pour les nœuds de feuille et deux paramètres ID (*linkID* et *linkID2*) pour les nœuds de branche (icônes fermée et ouverte). Pour les nœuds de feuille, le second paramètre est ignoré. Pour les nœuds de branche, si vous omettez *linkID2*, l'icône est utilisée pour les états fermé et ouvert.

Exemple

L'exemple suivant ajoute deux nœuds à une occurrence de composant Tree appelée `my_tr` et appelle la fonction `setIcon()` pour indiquer une icône dans la bibliothèque avec un identifiant de liaison `imageIcon` pour le deuxième nœud.

Vous devez d'abord ajouter une occurrence du composant Tree sur la scène et la nommer `my_tr` et ajouter une icône à la bibliothèque avec un identifiant de liaison `imageIcon` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Composant Tree sur la scène (nom d'occurrence : my_tr)
  - élément de bibliothèque avec identificateur de liaison de imageIcon
*/
```

```
var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0' /><node label='Outbox' data='1' /></node><node
  label='2nd Local Folders'><node label='Inbox' data='2' /><node
  label='Outbox' data='3' /></node>");
my_tr.dataProvider = trDP_xml;

// Définition de movieclip comme icône pour le 2nd nœud.
my_tr.setIcon(my_tr.getTreeNodeAt(1), "imageIcon");
```

Tree.setIsBranch()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
treeInstance.setIsBranch(node, isBranch)
```

Paramètres

node Nœud XML.

isBranch Valeur booléenne indiquant si le nœud est une branche (`true`) ou non (`false`).

Renvoie

Aucune.

Description

Méthode : indique si le nœud possède une icône de dossier et une flèche d'agrandissement et s'il a ou peut avoir des enfants. Un nœud est automatiquement défini en tant que branche lorsqu'il a des enfants ; il vous suffit d'appeler la méthode `setIsBranch()` pour créer un dossier vide. Vous pouvez également créer des branches n'ayant pas encore d'enfants : cela peut être utile si vous souhaitez par exemple que le chargement de nœuds enfant soit déclenché par l'ouverture d'un dossier par l'utilisateur.

L'appel à la méthode `setIsBranch()` permet d'actualiser tous les affichages.

Exemple

L'exemple suivant ajoute un seul nœud à une occurrence de composant Tree appelée `my_tr` et appelle `setIsBranch()` pour en faire une branche sans enfants.

Vous devez d'abord ajouter une occurrence du composant Tree sur la scène et la nommer `my_tr` ; ajoutez ensuite le code suivant à l'image 1.

```
/**
 Requiert :
  - Composant Tree sur la scène (nom d'occurrence : my_tr)
*/

var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label='Inbox' data='0'/>");
my_tr.dataProvider = trDP_xml;

// Définition du 1er nœud comme branche.
my_tr.setIsBranch(my_tr.getTreeNodeAt(0), true);
```

Tree.setIsOpen()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`treeInstance.setIsOpen(`*`node`*`, `*`open`*` [, `*`animate`*` [, `*`fireEvent`*`]])`

Paramètres

`node` Nœud XML.

`open` Valeur Booléenne qui ouvre un nœud (`true`) ou le ferme (`false`).

`animate` Valeur Booléenne qui détermine si la transition d'ouverture est animée (`true`) ou non (`false`). Ce paramètre est facultatif.

`fireEvent` Valeur Booléenne qui détermine si les événements `nodeOpen` et `nodeClose` sont distribués (`true`) ou non (`false`) lorsque le nœud de l'arborescence est ouvert ou fermé. Ce paramètre est facultatif. La valeur par défaut est `false`.

Valeur renvoyée

Aucune.

Description

Méthode : ouvre ou ferme un nœud.

Exemple

L'exemple suivant crée deux nœuds dans une arborescence de composant Tree appelée `my_tr` et appelle la méthode `setIsOpen()` pour ouvrir le deuxième nœud.

```
/**
 Requiert :
  - Composant Tree sur la scène (nom d'occurrence : my_tr)
*/
var my_tr:mx.controls.Tree;

my_tr.setSize(200, 100);

var trDP_xml:XML = new XML("<node label='1st Local Folders'><node
  label='Inbox' data='0'/><node label='Outbox' data='1'/></node><node
  label='2nd Local Folders'><node label='Inbox' data='2'/><node
  label='Outbox' data='3'/></node>");
my_tr.dataProvider = trDP_xml;

// Définition du 2ème nœud ouvert.
"my_tr.setIsOpen(my_tr.getTreeNodeAt(1), true);
```

CHAPITRE 51

Classe Tween

51

Héritage (classe Root)

Nom de classe ActionScript mx.transitions.Tween

La classe Tween vous permet en toute simplicité de déplacer, de redimensionner et de réaliser un fondu de clips sur la scène grâce à ActionScript, en définissant une propriété du clip cible afin qu'il soit animé par interpolation sur un nombre d'images ou de secondes.

REMARQUE	La classe Tween est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Cette classe vous permet également de spécifier de nombreuses méthodes d'accélération. L'*accélération* fait référence à l'accélération ou à la décélération progressive pendant une animation, ce qui lui donne un aspect plus réaliste. Par exemple, les options d'un composant de liste déroulante que vous créez peuvent augmenter progressivement leur vitesse au début d'une animation lorsqu'elles apparaissent, mais ralentir avant de s'arrêter complètement à la fin de l'animation lorsque la liste est étendue. Flash fournit de nombreuses méthodes qui contiennent des équations correspondant à cette accélération et à cette décélération, qui modifient l'animation d'accélération en conséquence.

La classe Tween appelle également des gestionnaires d'événements pour que votre code puisse répondre lorsqu'une animation démarre, s'arrête, reprend ou s'incrémente de la valeur de sa propriété interpolée. Par exemple, vous pouvez démarrer une seconde animation interpolée lorsque la première appelle le gestionnaire d'événements `Tween.onMotionStopped`, ce qui indique que la première interpolation est arrêtée.

Récapitulatif des méthodes de la classe Tween

Le tableau suivant répertorie les méthodes de la classe Tween.

Méthode	Description
`Tween.continueTo()`	Indique à l'animation interpolée de continuer de sa valeur actuelle à une nouvelle valeur.
`Tween.fforward()`	Transmet l'animation interpolée directement à la fin de l'animation.
`Tween.nextFrame()`	Transmet l'animation interpolée à l'image suivante.
`Tween.prevFrame()`	Dirige l'animation interpolée à l'image précédant l'image actuelle.
`Tween.resume()`	Reprend une animation interpolée de son point d'arrêt dans l'animation.
`Tween.rewind()`	Rembobine une animation interpolée à son début.
`Tween.start()`	Démarre l'animation interpolée du début.
`Tween.stop()`	Arrête l'animation interpolée sur sa position actuelle.
`Tween.toString()`	Renvoie le nom de classe, « [Tween] ».
`Tween.yoyo()`	Indique à l'animation interpolée d'effectuer la lecture à l'envers en partant de la dernière direction des incréments de propriété interpolés.

Récapitulatif des propriétés de la classe Tween

Le tableau suivant présente les propriétés de la classe Tween.

Propriété	Description
`Tween.duration`	La durée de l'animation interpolée en images ou en secondes. Lecture seule.
`Tween.finish`	La dernière valeur interpolée pour la fin de l'animation interpolée. Lecture seule.
`Tween.FPS`	Le nombre d'images par seconde de l'animation interpolée. Lecture seule.
`Tween.position`	La valeur actuelle de la propriété du clip cible en cours d'interpolation. Lecture seule.
`Tween.time`	L'heure actuelle dans la durée de l'animation. Lecture seule.

Récapitulatif des gestionnaires d'événements de la classe Tween

Le tableau suivant répertorie les gestionnaires d'événements de la classe Tween.

Evénement	Description
`Tween.onMotionChanged`	Gestionnaire d'événements : appelé lors de chaque changement d'une propriété de l'objet interpolé animé.
`Tween.onMotionFinished`	Gestionnaire d'événements ; appelé lorsque l'objet Tween termine son animation.
`Tween.onMotionResumed`	Gestionnaire d'événements : invoqué lors de l'appel à la méthode `Tween.resume()`, provoquant ainsi la reprise de l'animation interpolée.
`Tween.onMotionStarted`	Gestionnaire d'événements : invoqué lors de l'appel à la méthode `Tween.start()`, provoquant ainsi le début de l'animation interpolée.
`Tween.onMotionStopped`	Gestionnaire d'événements : invoqué lors de l'appel à la méthode `Tween.stop()`, provoquant ainsi l'arrêt de l'animation interpolée.

Utilisation de la classe Tween

Pour utiliser les méthodes et les propriétés de la classe Tween, utilisez l'opérateur `new` pour créer une occurrence de la classe. Par exemple, pour appliquer une occurrence d'interpolation à un objet clip sur la scène appelé `myMovieClip_mc`, utilisez le code suivant pour créer une occurrence `mx.transitions.Tween` :

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(myMovieClip_mc, "_x",
  mx.transitions.easing.Elastic.easeOut, 0, 300, 3, true);
```

Paramètres de la classe Tween

Lorsque vous créez une occurrence d'une classe Tween, vous transmettez plusieurs paramètres. Vous devez indiquer l'objet clip cible, la propriété du clip que l'interpolation doit affecter, la plage sur laquelle l'objet doit être interpolé et la méthode d'accélération à utiliser pour calculer la propriété interpolée

Le constructeur de la classe mx.transitions.Tween possède les noms et les types de paramètres suivants :

```
Tween( obj:Object, prop:String, func:Function, begin:Number, finish:Number,
  duration:Number, useSeconds:Boolean )
```

obj Objet clip ciblé par l'occurrence Tween.

prop Nom de type chaîne d'une propriété dans `obj` dans laquelle les valeurs doivent être interpolées.

func Méthode d'accélération qui calcule un effet d'accélération pour les valeurs de propriété de l'objet interpolé. Voir « Présentation des classes et des méthodes d'accélération », à la page 1366.

begin Nombre indiquant la valeur de départ de `prop` (propriété de l'objet cible à interpoler).

finish Nombre indiquant la valeur de fin de `prop` (propriété de l'objet cible à interpoler).

duration Nombre indiquant la durée du mouvement d'interpolation. Si ce paramètre est omis, la durée est définie sur l'infini par défaut.

useSeconds Valeur booléenne indiquant d'utiliser des secondes si elle est définie sur `true` ou des images si elle est définie sur `false` par rapport à la valeur définie dans le paramètre `duration`.

Présentation des classes et des méthodes d'accélération

Pour créer une occurrence de la classe Tween, utilisez le paramètre `func` pour indiquer une fonction ou une méthode qui fournit un calcul d'accélération. Flash propose cinq classes d'accélération, chacune avec trois méthodes qui indiquent la partie du mouvement de transition à laquelle appliquer l'effet d'accélération : au début de l'animation, la fin ou les deux. Par ailleurs, une classe d'accélération `None` pourvue d'une méthode `easeNone` permet d'indiquer qu'aucune accélération n'est utilisée.

Les classes et les composants suivants utilisent les classes et les méthodes d'accélération :

- La classe mx.transitions.Tween pour les effets d'accélération sur une animation interpolée.
- La classe mx.transitions.TransitionManager pour les effets d'accélération sur des transitions. Voir Chapitre 48, « Classe TransitionManager », à la page 1285.
- Certains composants. Voir « Application des méthodes d'accélération aux composants », à la page 1368.

Les six classes de calcul d'accélération sont décrites dans le tableau suivant :

Classe d'accélération	Description
Back	Etend l'animation au-delà de la plage de transition à l'une ou aux deux extrémités pour créer un effet de retrait.
Bounce	Ajoute un effet de rebond à la plage de transition, à l'une ou aux deux extrémités. Le nombre de rebonds est en rapport avec la durée : plus la durée est longue, plus le nombre de rebonds produits est important.
Elastic	Ajoute un effet élastique qui sort de la plage de transition à l'une ou aux deux extrémités. Le taux d'élasticité n'est pas affecté par la durée.
Regular	Ajoute un mouvement plus lent à l'une ou aux deux extrémités. Cette fonction vous permet d'ajouter un effet d'accélération, de ralentissement, ou les deux.
Strong	Ajoute un mouvement plus lent à l'une ou aux deux extrémités. Cet effet est similaire à la classe d'accélération Regular, mais il est plus prononcé.
None	Ajoute un mouvement régulier du début à la fin, sans effet, ralentissement ni accélération. Cette transition est également appelée une transition linéaire.

Chacune de ces six classes de calcul d'accélération a trois méthodes d'accélération qui indiquent à quelle partie de l'animation l'effet d'accélération doit être appliqué. Par ailleurs, la classe d'accélération None possède une quatrième méthode d'accélération : easeNone. Les méthodes d'accélération sont décrites dans le tableau ci-dessous.

Méthode	Description
easeIn	Produit un effet d'accélération au début de la transition.
easeOut	Produit un effet d'accélération à la fin de la transition.
easeInOut	Produit un effet d'accélération au début et à la fin de la transition.
easeNone	Indique qu'aucun calcul d'accélération ne doit être utilisé. Fournie uniquement dans la classe d'accélération None.

Application des méthodes d'accélération aux composants

Une autre utilisation des différentes méthodes d'accélération consiste à les appliquer à certains composants. Vous pouvez appliquer les méthodes d'accélération uniquement aux composants suivants : Accordion, ComboBox, DataGrid, List, Menu et Tree. Chaque composant utilise les méthodes d'accélération pour permettre différentes personnalisations. Par exemple, les composants Accordion, ComboBox et Tree vous permettent de sélectionner une classe d'accélération à utiliser pour leurs animations d'ouverture et de fermeture respectives. Le composant Menu, en revanche, vous permet de définir uniquement la durée (en millisecondes) de l'animation.

Application des méthodes d'accélération à un composant Accordion

Cette section décrit comment ajouter un composant Accordion à un composant Flash, ajouter des diapositives enfants et modifier la méthode d'accélération par défaut ainsi que la durée. Si vous décidez d'utiliser ce code dans un projet, réduisez la valeur de la propriété `openDuration` afin de ne pas ennuyer les utilisateurs avec des animations trop lentes lorsqu'ils ouvrent et ferment les panneaux enfants du composant Accordion.

Pour appliquer une méthode d'accélération différente au composant Accordion :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Enregistrez le fichier sous le nom accordion.fla.
3. Faites glisser une copie du composant Accordion sur la scène.
4. Ouvrez l'inspecteur Propriétés, puis tapez `my_acc` dans la zone de texte Nom de l'occurrence.
5. Insérez un nouveau calque au-dessus du calque 1 et nommez-le *actions*.

6. Ajoutez le code ActionScript suivant dans l'image 1 du calque actions :

```
import mx.core.View;
import mx.transitions.easing.*;
my_acc.createChild(View, "studio_view", {label:"Studio"});
my_acc.createChild(View, "dreamweaver_view", {label:"Dreamweaver"});
my_acc.createChild(View, "flash_view", {label:"Flash"});
my_acc.createChild(View, "coldfusion_view", {label:"ColdFusion"});
my_acc.createChild(View, "contribute_view", {label:"Contribute"});
my_acc.setStyle("openEasing", Bounce.easeOut);
my_acc.setStyle("openDuration", 3500);
```

   Ce code importe les classes d'accélération. Vous pouvez donc taper `Bounce.easeOut` au lieu de vous référer à chacune des classes avec leurs noms complets, tels que `mx.transitions.easing.Bounce.easeOut`. Le code ajoute ensuite cinq nouveaux panneaux enfants au composant Accordion (Studio, Dreamweaver, Flash, ColdFusion et Contribute). Les deux dernières lignes de code définissent le style d'accélération de la méthode d'accélération par défaut sur `Bounce.easeOut`, et définissent la longueur de l'animation sur 3 500 millisecondes (3,5 secondes).

7. Enregistrez le document, puis choisissez Contrôle > Tester l'animation pour prévisualiser le fichier dans l'environnement de test.

   Cliquez sur chacune des différentes barres d'en-tête (titre) pour afficher les animations modifiées et passer d'un panneau à l'autre.

   Si vous souhaitez augmenter la vitesse de l'animation, diminuez le paramètre `openDuration` en le faisant passer de 3 500 millisecondes à une valeur inférieure. La durée par défaut de l'animation est définie sur 250 millisecondes (un quart de seconde).

Application des méthodes d'accélération au composant ComboBox

La procédure de modification de la méthode d'accélération par défaut dans un composant ComboBox est identique à l'exemple « Application des méthodes d'accélération à un composant Accordion », à la page 1368 dans lequel vous modifiez l'animation du composant Accordion. Dans l'exemple suivant, vous utilisez ActionScript pour ajouter dynamiquement le composant sur la scène à l'exécution.

Pour appliquer des méthodes d'accélération à un composant ComboBox :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Enregistrez le fichier sous le nom combobox.fla.
3. Faites glisser une copie du composant ComboBox du panneau Composants à la bibliothèque du document actuel.

REMARQUE	Le composant apparaît dans la bibliothèque (et non sur la scène). Il est disponible pour le fichier SWF à l'exécution.

4. Insérez un nouveau calque et renommez-le actions.

   Assurez-vous que le calque actions se trouve au-dessus du Calque 1.

5. Ajoutez le code ActionScript suivant dans l'image 1 du calque actions :

```
import mx.transitions.easing.*;
this.createClassObject(mx.controls.ComboBox, "my_cb", 20);
var product_array:Array = new Array("Studio", "Dreamweaver", "Flash",
  "ColdFusion", "Contribute", "Breeze", "Director", "Flex");
my_cb.dataProvider = product_array;
my_cb.move(10, 10);
my_cb.setSize(140, 22);
my_cb.setStyle("openDuration", 2000);
my_cb.setStyle("openEasing", Elastic.easeOut);
```

   Une fois que vous avez importé chacune des méthodes d'accélération (dans la première ligne de code), la méthode `createClassObject()` crée une occurrence du composant ComboBox. Le mot-clé `this` à la deuxième ligne de code fait référence au scénario principal du fichier SWF. Cette ligne de code place le composant sur la scène à l'exécution et lui donne le nom d'occurrence `my_cb`.

   Vous créez ensuite un tableau appelé `product_array` qui contient une liste des logiciels Adobe. Vous utilisez ce tableau dans la ligne de code suivante pour définir la propriété `dataProvider` sur le tableau des noms de produit. Vous utilisez ensuite la méthode `setSize()` pour redimensionner l'occurrence de composant, définissez le paramètre `openDuration` sur 2 000 millisecondes (2 secondes), puis modifiez la méthode d'accélération sur `Elastic.easeOut`.

REMARQUE	A l'instar des exemples précédents, vous importez les classes d'accélération, ce qui vous permet d'utiliser la version raccourcie du nom de classe au lieu du nom de classe complet `mx.transitions.easing.Elastic.easeOut`.

6. Enregistrez le document actuel, puis choisissez Contrôle > Tester l'animation pour afficher le document dans l'environnement de test.
7. Cliquez sur le composant ComboBox sur la scène pour animer la liste déroulante de noms de produit à l'aide de la classe d'accélération spécifiée.

REMARQUE	Soyez vigilant lorsque vous utilisez une méthode d'accélération telle que Elastic ou Bounce pour vos composants ComboBox ou Accordion. Certains utilisateurs risquent d'être ennuyés si vos options mettent du temps à arrêter de se déplacer avant qu'ils puissent les lire et les sélectionner dans le menu. Testez vos applications et paramètres et voyez si les méthodes d'accélération améliorent ou non votre document Flash.

Animation du composant DataGrid

Flash vous permet également de régler les animations que vous utilisez lorsque vous sélectionnez des éléments dans un composant (les composants DataGrid, Tree, ComboBox ou List, par exemple). Même si les animations sont subtiles, dans certains cas, il est utile de contrôler de petits détails ou d'augmenter la vitesse de l'animation.

Pour ajouter une accélération au composant DataGrid :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Enregistrez le fichier sous le nom datagrid.fla.
3. Faites glisser une occurrence du composant DataGrid sur la scène et nommez l'occurrence `my_dg`.
4. Insérez un nouveau calque et renommez-le actions.

   Assurez-vous que le calque actions se trouve au-dessus de Calque 1.
5. Ajoutez le code ActionScript suivant au calque actions :

```
import mx.transitions.easing.*;
my_dg.setSize(320, 240);
my_dg.addColumn("product");
my_dg.getColumnAt(0).width = 304;
my_dg.rowHeight = 60;
my_dg.addItem({product:"Studio"});
my_dg.addItem({product:"Dreamweaver"});
my_dg.addItem({product:"Flash"});
my_dg.setStyle("selectionEasing", Elastic.easeInOut);
my_dg.setStyle("selectionDuration", 1000);
```

Ce code ActionScript importe les classes d'accélération et redimensionne l'occurrence de composant sur la scène sur 320 pixels (largeur) par 240 pixels (hauteur). Créez ensuite une colonne appelée *product* et redimensionnez-la sur 304 pixels (largeur). La grille de données a une largeur de 320 pixels alors que la barre de défilement a une largeur de 16 pixels, ce qui laisse une différence de 304 pixels. Définissez ensuite la hauteur de ligne sur 60 pixels afin que les animations d'accélération soient plus faciles à visualiser.

Les trois lignes suivantes de code ActionScript ajoutent des éléments à l'occurrence de la grille de données pour vous permettre de cliquer et de visualiser les animations. Pour finir, définissez les propriétés `selectionEasing` et `selectionDuration` à l'aide de la méthode `setStyle()`. Définissez la méthode d'accélération sur `Elastic.easeInOut` et le paramètre `duration` sur 1 000 millisecondes (une seconde, ce qui est cinq fois plus long que la valeur par défaut de 200 millisecondes).

6. Enregistrez le document et choisissez Contrôle > Tester l'animation pour afficher le résultat dans l'environnement de test.

   Lorsque vous cliquez sur un élément dans l'occurrence de composant DataGrid, la sélection accélère et décélère à l'aide de l'effet élastique. L'animation est facile à visualiser car la durée est augmentée de façon significative.

REMARQUE

Vous pouvez également utiliser les mêmes propriétés (`selectionEasing` et `selectionDuration`) avec les composants ComboBox, List et Tree.

Tween.continueTo()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
tweenInstance.continueTo(finish, duration)
```

Paramètres

finish Nombre indiquant la valeur de fin de la propriété de l'objet cible à interpoler.

duration Nombre indiquant la durée ou le nombre d'images pour le mouvement d'interpolation ; la durée est mesurée en termes de temps si le paramètre `Tween.start()` `useSeconds` est défini sur true ou en termes d'image s'il est défini sur false. Pour plus d'informations sur le paramètre `useSeconds`, reportez-vous à « Tween.start() » à la page 1388.

Valeur renvoyée

Aucune.

Description

Méthode : indique à l'animation interpolée de poursuivre l'interpolation de son point d'animation actuel à un nouveau point de durée et de fin.

Exemple

Dans cet exemple, un gestionnaire est déclenché par l'événement `onMotionFinished` et indique à une occurrence Tween de continuer son animation avec de nouvelles valeurs `finish` et `duration` en appelant la méthode `Tween.continueTo()` : Une occurrence de clip nommée `img1_mc` est requise sur la scène pour cet exemple.

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_y",
  mx.transitions.easing.Elastic.easeOut,0, 200, 3, true);
myTween.onMotionFinished = function() {
  var myFinish:Number = 100;
  var myDuration:Number = 5;
  myTween.continueTo( myFinish, myDuration );
};
```

Tween.duration

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

tweenInstance.duration

Description

Propriété (lecture seule) : nombre indiquant la durée de l'animation interpolée en images ou en secondes. Cette propriété est définie comme un paramètre lors de la création d'une occurrence Tween ou de l'appel à la méthode `Tween.yoyo()`.

Exemple

L'exemple suivant recherche le paramètre `duration` actuel d'un objet Tween en accédant à la propriété `Tween.duration`. Une occurrence de clip nommée `img1_mc` est requise sur la scène pour cet exemple.

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_y",
  mx.transitions.easing.Strong.easeOut,0, Stage.height, 50, false);
var theDuration:Number = myTween.duration;
trace(theDuration);
```

Tween.fforward()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`tweenInstance``.fforward()`

Valeur renvoyée

Aucune.

Description

Méthode : transmet l'animation interpolée directement à la valeur finale de l'animation interpolée.

Exemple

Dans cet exemple, la méthode `Tween.fforward()` est appelée pour transmettre une animation interpolée directement à sa valeur finale, déclenchant immédiatement l'événement `onMotionFinished`. Un gestionnaire d'événements `Tween.onMotionFinished` appelle la méthode `Tween.yoyo()`. Par conséquent, l'animation interpolée démarre de façon visible avec l'effet inversé de la méthode `Tween.yoyo()`, étant donné que l'animation initiale a été ignorée pour passer à sa fin. Une occurrence de clip nommée `img1_mc` est requise sur la scène pour cet exemple.

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.Elastic.easeOut,0, Stage.width - img1_mc._width, 8,
  true);

myTween.fforward();

myTween.onMotionFinished = function() {
  myTween.yoyo();
};
```

Tween.finish

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

tweenInstance.finish

Description

Propriété (lecture seule) : nombre indiquant la valeur de fin de la propriété de l'objet cible à interpoler. Cette propriété est définie comme un paramètre lors de la création d'une occurrence Tween ou de l'appel à la méthode `Tween.yoyo()`.

Exemple

L'exemple suivant renvoie le paramètre `finish` actuel d'une occurrence Tween.
Une occurrence de clip nommée `img1_mc` est requise sur la scène pour cet exemple.

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_y",
  mx.transitions.easing.Strong.easeOut,0, Stage.height - img1_mc._height,
  50, false);
var myFinish:Number = myTween.finish;
trace(myFinish);
```

Tween.FPS

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`tweenInstance``.FPS`

Description

Propriété : nombre d'images par seconde calculé dans l'animation interpolée. Par défaut, la fréquence d'images actuelle de la scène est utilisée pour calculer l'animation interpolée. Définir cette propriété permet de recalculer le nombre d'incréments dans la propriété animée, qui est affichée chaque seconde, sur la propriété `Tween.FPS` plutôt que sur la fréquence d'images actuelle de la scène. La définition de la propriété `Tween.FPS` ne modifie pas la fréquence d'images effective de la scène.

REMARQUE	La propriété Tween.FPS renvoie undefined à moins que vous ne l'ayez définie en premier de façon explicite.

Exemple

L'exemple suivant crée deux animations interpolées définies sur deux paramètres différents `FPS`. Les paramètres `FPS` actuels des deux occurrences Tween s'affichent. Une occurrence de clip nommée `img1_mc` et une autre occurrence de clip nommée `img2_mc` sont requises sur la scène pour cet exemple.

```
import mx.transitions.Tween;
var myTween1:Tween = new Tween(img1_mc, "_y",
  mx.transitions.easing.Strong.easeOut,0, Stage.height - img1_mc._height,
  400, false);
myTween1.FPS = 1;
var myFPS1:Number = myTween1.FPS;
trace("myTween1.FPS:" + myFPS1);

var myTween2:Tween = new Tween(img2_mc, "_y",
  mx.transitions.easing.Strong.easeOut,0, Stage.height - img2_mc._height,
  400, false);
myTween2.FPS = 12;
var myFPS2:Number = myTween2.FPS;
trace("myTween2.FPS:" + myFPS2);
```

Tween.nextFrame()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`tweenInstance``.nextFrame()`

Valeur renvoyée

Aucune.

Description

Méthode : transmet l'animation interpolée à l'image suivante d'une animation arrêtée. Utilisez cette méthode pour transmettre une image à la fois d'une animation interpolée une fois que vous l'avez arrêtée à l'aide de la méthode `Tween.stop()`.

REMARQUE	Cette méthode peut être utilisée uniquement sur des interpolations basées sur image. Une interpolation est basée sur image lors de sa création en définissant le paramètre `useSeconds` sur false.

Exemple

Cet exemple applique une animation interpolée au clip `img1_mc`. L'animation est lue en boucle et se répète à partir de son point de départ en appelant la méthode `Tween.start()` depuis un gestionnaire déclenché par l'événement `Tween.onMotionFinished`. Cliquez sur un bouton appelé forwardByFrame_btn pour appeler la méthode `Tween.stop()` et arrêter l'animation, puis appelez la méthode `Tween.nextFrame()`. Lorsque vous cliquez sur le bouton pendant l'animation interpolée, l'animation est arrêtée, puis avance image par image. Lorsque vous créez l'occurrence Tween, le paramètre `useSeconds` est déclaré false pour que l'interpolation soit basée sur image. Cette procédure est requise pour utiliser la méthode `Tween.nextFrame()`. Une occurrence de clip nommée `img1_mc` est requise sur la scène pour cet exemple.

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone,0, Stage.width, 60, false);

myTween.onMotionFinished = function() {
  myTween.start();
};

forwardByFrame_btn.onRelease = function() {
   myTween.stop();
   myTween.nextFrame();
};
```

Tween.onMotionChanged

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
tweenInstance.onMotionChanged = function() {
  // ...
};
```

Description

Gestionnaire d'événements : appelé lors de chaque changement d'une propriété de l'objet interpolé animé. La gestion de cet événement permet à votre code de réagir lorsque la propriété du clip cible en cours d'interpolation est incrémentée à la valeur suivante.

Exemple

Cet exemple applique une interpolation au clip `img1_mc`. Chaque incrément de l'interpolation étant appliqué à la propriété `_x` du clip, le gestionnaire d'événements `onMotionChanged` est appelé et affiche un message de suivi indiquant la nouvelle position du clip interpolé. Une occurrence de clip nommée `img1_mc` est requise sur la scène pour cet exemple.

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.Elastic.easeOut,0, Stage.width-img1_mc._width, 3,
  true);

myTween.onMotionChanged = function() {
  trace( this.position );
};
```

Tween.onMotionFinished

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
tweenInstance.onMotionFinished = function() {
  // ...
};
```

Description

Gestionnaire d'événements : appelé lorsque l'animation atteint la fin de sa durée. La gestion de cet événement permet à votre code de réagir au point de fin de l'animation interpolée.

Exemple

L'exemple suivant applique une interpolation au clip `img1_mc`. Lorsque l'interpolation atteint la fin de son animation, elle appelle le gestionnaire d'événements `onMotionFinished` qui appelle la méthode `Tween.yoyo()`. L'interpolation est donc capable de terminer son animation avant que la méthode `Tween.yoyo()` soit appelée pour inverser l'animation. Une occurrence de clip nommée `img1_mc` est requise sur la scène pour cet exemple.

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.Elastic.easeOut,0, Stage.width-img1_mc._width, 3,
  true);
myTween.FPS = 30;
myTween.onMotionFinished = function() {
  myTween.yoyo();
};
```

Tween.onMotionResumed

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
tweenInstance.onMotionResumed = function() {
  // ...
};
```

Description

Gestionnaire d'événements ; invoqué à l'appel de la méthode `Tween.resume()`. La gestion de cet événement permet à votre code de réagir au point de reprise de l'animation interpolée.

Exemple

L'exemple suivant applique une animation interpolée au clip `img1_mc`. L'animation est lue en boucle et se répète à partir de son point de départ en appelant la méthode `Tween.start()` depuis un gestionnaire d'événements `onMotionFinished`. Cliquez sur un bouton appelé `stopTween_btn` pour appeler la méthode `Tween.stop()` et arrêter l'animation interpolée à sa valeur actuelle. Cliquez sur un bouton appelé resumeTween_btn pour appeler la méthode `Tween.resume()` et reprendre l'animation interpolée depuis son point d'arrêt. Lorsque la méthode `Tween.resume()` est appelée, l'occurrence Tween appelle le gestionnaire `onMotionResumed`. Les trois occurrences de clip nommées `img1_mc`, `stopTween_btn` et `resumeTween_btn` sont requises sur la scène pour cet exemple.

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone,0, Stage.width, 3, true);

myTween.onMotionFinished = function() {
  myTween.start();
};

myTween.onMotionResumed = function() {
  trace("onMotionResumed");
};

stopTween_btn.onRelease = function() {
   myTween.stop();
};

resumeTween_btn.onRelease = function() {
   myTween.resume();
};
```

Tween.onMotionStarted

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
tweenInstance.onMotionStarted = function() {
  // ...
};
```

Description

Evénement : appelé lorsque l'animation redémarre pendant ou à la fin de son animation. Ce gestionnaire d'événements n'est pas appelé au lancement initial d'une animation interpolée. Appeler la méthode `Tween.start()`, `Tween.yoyo()` ou `Tween.yoyo()` pour redémarrer une animation terminée ou une animation en cours permet d'appeler le gestionnaire d'événements `onMotionStarted`. La gestion de cet événement permet à votre code de réagir au point de redémarrage de l'animation d'interpolée après son lancement initial.

Exemple

L'exemple suivant applique une animation interpolée au clip `img1_mc`. L'animation est lue en boucle et se répète à partir de son point de départ en appelant la méthode `Tween.start()` depuis le gestionnaire d'événements `Tween.onMotionFinished`. Lorsque la méthode `Tween.start()` est appelée, l'occurrence Tween appelle le gestionnaire d'éviments `Tween.onMotionStarted`. Une occurrence de clip nommée `img1_mc` est requise sur la scène pour cet exemple.

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone,0, Stage.width, 4, true);

myTween.onMotionFinished = function() {
  myTween.start();
};

myTween.onMotionStarted = function() {
  trace("onMotionStarted");
};
```

Tween.onMotionStopped

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
tweenInstance.onMotionStopped = function() {
  // ...
};
```

Description

Evénement : appelé lorsque l'animation interpolée est effectuée jusqu'à la fin de son animation ou lorsque la méthode `Tween.stop()` est appelée. La gestion de cet événement permet à votre code de réagir au point d'arrêt de l'animation interpolée.

Exemple

L'exemple suivant applique une animation interpolée au clip `img1_mc`. Lorsque l'occurrence Tween termine son animation, elle appelle le gestionnaire d'événements `Tween.onMotionStopped`. Une occurrence de clip nommée `img1_mc` est requise sur la scène pour cet exemple.

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone,0, Stage.width-img1_mc._width, 3,
  true);

myTween.onMotionStopped = function() {
  trace("onMotionStopped");
};
```

Tween.position

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

tweenInstance.position

Description

Propriété (lecture seule) : valeur actuelle de la propriété de l'objet cible en cours d'interpolation. Cette valeur est mise à jour avec chaque image dessinée de l'animation interpolée.

Exemple

L'exemple suivant recherche le paramètre `Tween.position` actuel d'un l'objet Tween et la valeur `Tween.position` à laquelle se situe la position de l'interpolation sur la dernière image de l'animation interpolée. Une occurrence de clip nommée `img1_mc` est requise sur la scène pour cet exemple.

```
import mx.transitions.Tween;
var myTween:Tween = new mx.transitions.Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone, 0, Stage.width-img1_mc._width, 3,
  true);
myTween.onMotionChanged = function() {
  var myPosition:Number = myTween.position;
  var myFinish:Number = myTween.finish;
  trace(myPosition + " : " + myFinish);
};
```

Tween.prevFrame()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

tweenInstance`.prevFrame()`

Valeur renvoyée

Aucune.

Description

Méthode : lit l'image précédente de l'animation interpolée depuis le point d'arrêt actuel d'une animation qui a été arrêtée. Utilisez cette méthode pour lire une animation interpolée en arrière d'une image à la fois après l'avoir arrêtée à l'aide de la méthode `Tween.stop()`.

REMARQUE

Cette méthode peut être utilisée uniquement sur des interpolations basées sur image. Une interpolation est basée sur image lors de sa création en définissant le paramètre `Tween.start()useSeconds` sur false. Pour plus d'informations sur le paramètre `useSeconds`, reportez-vous à « Tween.start() » à la page 1388.

Exemple

Cet exemple applique une animation interpolée au clip `img1_mc`. L'animation est lue en boucle et se répète à partir de son point de départ en appelant la méthode `Tween.start()` depuis un gestionnaire déclenché par l'événement `onMotionFinished`. Cliquez sur un bouton appelé forwardByFrame_btn pour appeler la méthode `Tween.stop()` et arrêter l'animation, puis appelez la méthode `Tween.prevFrame()`. Cliquer sur le bouton pendant l'animation interpolée arrête cette dernière, puis reprend sa lecture en sens inverse image par image. Cliquer sur le bouton resumeTween_btn appelle la méthode Tween.resume(), et l'animation interpolée reprend. Lorsque vous créez l'occurrence Tween, le paramètre `useSeconds` est défini sur `false` pour que l'interpolation soit basée sur image. Cette procédure est requise pour utiliser la méthode `Tween.nextFrame()`. Les trois occurrences de clip nommées `img1_mc`, `resumeTween_btn` et `reverseByFrame_btn` sont requises sur la scène pour cet exemple.

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone, -img1_mc._width, Stage.width, 50,
  false);

myTween.onMotionFinished = function() {
  myTween.start();
};

reverseByFrame_btn.onRelease = function() {
   myTween.stop();
   myTween.prevFrame();
};
resumeTween_btn.onRelease = function() {
   myTween.resume();
};
```

Tween.resume()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

tweenInstance.resume()

Valeur renvoyée

Aucune.

Description

Méthode : reprend la lecture d'une animation interpolée arrêtée. Utilisez cette méthode pour continuer une animation interpolée une fois que vous l'avez arrêtée à l'aide de la méthode `Tween.stop()`.

REMARQUE	Cette méthode peut être utilisée uniquement sur des interpolations basées sur image. Une interpolation est basée sur image lors de sa création en définissant le paramètre `useSeconds` sur false.

Exemple

Cet exemple applique une animation interpolée au clip `img1_mc`. L'animation est lue en boucle et se répète à partir de son point de départ en appelant la méthode `Tween.start()` depuis un gestionnaire déclenché par l'événement `onMotionFinished`. Cliquez sur le bouton stopTween_btn pour appeler la méthode `Tween.stop()` et arrêter l'animation interpolée à sa valeur actuelle. Cliquez sur le bouton resumeTween_btn pour appeler la méthode `Tween.resume()` et reprendre l'animation interpolée depuis son point d'arrêt. Les trois occurrences de clip nommées `img1_mc`, `stopTween_btn` et `resumeTween_btn` sont requises sur la scène pour cet exemple.

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone, -img1_mc._width, Stage.width, 3,
  true);

myTween.onMotionFinished = function() {
  myTween.start();
};

stopTween_btn.onRelease = function() {
   myTween.stop();
};
resumeTween_btn.onRelease = function() {
   myTween.resume();
};
```

Tween.rewind()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

tweenInstance.rewind()

Paramètres

Aucun.

Valeur renvoyée

Aucune.

Description

Méthode : recule la lecture d'une animation interpolée jusqu'à sa valeur de départ. Si la méthode `Tween.rewind()` est appelée pendant la lecture de l'animation interpolée, l'animation est rembobinée jusqu'à sa valeur de départ et poursuit la lecture. Si la méthode `Tween.rewind()` est appelée alors que l'animation interpolée est arrêtée ou a terminé son animation, l'animation interpolée est rembobinée jusqu'à sa valeur de départ, et reste arrêtée. Utilisez cette méthode pour rembobiner une animation interpolée à son point de départ une fois que vous l'avez arrêtée en utilisant la méthode `Tween.stop()` ou pour rembobiner une animation interpolée pendant sa lecture.

Exemple

L'exemple suivant applique une animation interpolée au clip `img1_mc`. L'animation est lue en boucle et se répète à partir de son point de départ en appelant la méthode `Tween.start()` depuis un gestionnaire déclenché par l'événement `Tween.onMotionFinished`. Cliquez sur le bouton resumeTween_btn pour appeler la méthode `Tween.resume()` et rembobiner l'animation interpolée jusqu'à son point de départ. Les quatre occurrences de clip nommées `img1_mc`, `stopTween_btn`, `rewindTween_btn` et `resumeTween_btn` sont requises sur la scène pour cet exemple.

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone, img1_mc._width, Stage.width, 8,
  true);

myTween.onMotionFinished = function() {
  myTween.start();
};

stopTween_btn.onRelease = function() {
  myTween.stop();
};

rewindTween_btn.onRelease = function() {
  myTween.rewind();
};

resumeTween_btn.onRelease = function() {
  myTween.resume();
};
```

Tween.start()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

tweenInstance`.start()`

Valeur renvoyée

Aucune.

Description

Méthode : démarre la lecture d'une animation interpolée depuis son point de départ. Cette méthode permet de redémarrer une occurrence Tween depuis le début de son animation après son arrêt ou la fin de son animation.

Exemple

Cet exemple crée un objet Tween qui anime la propriété `_x` du clip `img1_mc`. Lorsque l'animation interpolée est terminée et appelle le gestionnaire d'événements `Tween.onMotionFinished`, l'interpolation est relue en appelant la méthode `Tween.start()`. Le clip obtenu se déplace à travers la scène de gauche à droite, puis lance de nouveau l'animation lorsqu'il atteint l'extrémité de la scène. Une occurrence de clip nommée `img1_mc` est requise sur la scène pour cet exemple.

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone,0, Stage.width, 50, false);

myTween.onMotionFinished = function() {
  myTween.start();
};
```

Tween.stop()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`tweenInstance``.stop()`

Valeur renvoyée

Aucune.

Description

Méthode : arrête la lecture d'une animation interpolée à sa valeur actuelle.

Exemple

L'exemple suivant applique une animation interpolée à la propriété _x du clip img1_mc. Un gestionnaire onRelease() du clip stopTween_btn appelle la méthode Tween.stop() pour arrêter l'animation interpolée, et un gestionnaire onRelease() du clip resumeTween_btn appelle la méthode Tween.resume() pour reprendre l'animation depuis son point d'arrêt. Les trois occurrences de clip nommées img1_mc, stopTween_btn et resumeTween_btn sont requises sur la scène pour cet exemple.

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone, img1_mc._width, Stage.width, 8,
  true);

myTween.onMotionFinished = function() {
  myTween.start();
};

stopTween_btn.onRelease = function() {
   myTween.stop();
};

resumeTween_btn.onRelease = function() {
   myTween.resume();
};
```

Tween.time

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
tweenInstance.time
```

Description

Propriété (lecture seule) : nombre actuel de secondes transmises dans la durée de l'animation si le paramètre useSeconds a été défini sur true lors de la création de l'occurrence Tween. Si ce paramètre a été défini sur false, Tween.time renvoie le nombre actuel d'images transmises dans l'animation de l'objet Tween.

Exemple

L'exemple suivant renvoie la valeur time d'une occurrence Tween. Une occurrence de clip nommée `img1_mc` est requise sur la scène pour cet exemple.

```
import mx.transitions.Tween;
var myTween:Tween = new mx.transitions.Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone, 0, Stage.width-img1_mc._width, 10,
  false);
myTween.onMotionChanged = function() {
  var myCurrentTime:Number = myTween.time;
  var myCurrentDuration:Number = myTween.duration;
trace(myCurrentTime + " of " + myCurrentDuration);
};
```

Tween.toString()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`tweenInstance``.toString()`

Valeur renvoyée

La chaîne suivante est renvoyée : « [Tween] ».

Description

Méthode : renvoie le nom de classe, « [Tween] ».

Exemple

Dans l'exemple suivant, un objet Tween est identifié suite à l'appel à la méthode `Tween.toString()` pour renvoyer « `[Tween]` », qui identifie le nom de classe de cet objet. Une occurrence de clip nommée `img1_mc` est requise sur la scène pour cet exemple.

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_alpha",
  mx.transitions.easing.None.easeNone,0, 100, 50, false);
var theClassName:String = myTween.toString();
trace(theClassName);
```

Tween.yoyo()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

tweenInstance.yoyo()

Valeur renvoyée

Aucune.

Description

Méthode : indique à l'animation interpolée d'effectuer la lecture à l'envers en partant de la dernière direction des incréments de propriété interpolés. Si cette méthode est appelée avant la fin de l'animation d'un objet Tween, l'animation passe directement à la fin de sa lecture, puis lit dans la direction inverse depuis ce point. Si vous appelez la méthode `Tween.yoyo()` dans un gestionnaire d'événements `Tween.onMotionFinished`, vous pouvez obtenir un effet d'animation qui termine sa lecture, puis inverse toute sa lecture. Cette procédure garantit que l'effet inverse de la méthode Tween.yoyo ne commence pas tant que l'animation interpolée actuelle n'est pas terminée. Voir « Tween.onMotionFinished » à la page 1379.

Exemple

Dans l'exemple suivant, un gestionnaire est déclenché par l'événement `Tween.onMotionFinished` et indique à l'occurrence Tween d'animer le clip `img1_mc` dans une direction inverse en appelant la méthode `Tween.yoyo()`. Le clip obtenu se déplace de gauche à droite sur la scène, puis inverse sa direction, en se déplaçant de gauche à droite dans une boucle d'animation. Une occurrence de clip nommée `img1_mc` est requise sur la scène pour cet exemple.

```
import mx.transitions.Tween;
var myTween:Tween = new Tween(img1_mc, "_x",
  mx.transitions.easing.None.easeNone,0, Stage.width, 4, true);
myTween.onMotionFinished = function() {
  myTween.yoyo();
};
```

CHAPITRE 52

Classe UIComponent

52

La classe UIComponent ne représente pas un composant visuel ; elle contient des méthodes, des propriétés et des événements qui permettent aux composants Adobe de partager des comportements communs.

REMARQUE	La classe UIComponent est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Tous les composants étendent UIComponent. La classe UIComponent vous permet d'effectuer les opérations suivantes :

- Recevoir le focus et la saisie au clavier
- Activer et désactiver des composants
- Redimensionner en fonction de la disposition

Pour utiliser les méthodes et les propriétés de UIComponent, appelez-les directement du composant que vous utilisez. Par exemple, pour appeler `UIComponent.setFocus()` à partir du composant RadioButton, écrivez le code suivant :

```
myRadioButton.setFocus();
```

Vous devez seulement créer une occurrence de UIComponent si vous créez un composant. Même dans ce cas, la classe UIComponent est toujours créée de façon implicite par d'autres sous-classes telles que Button. Si vous n'avez pas besoin de créer une occurrence de la classe UIComponent, utilisez le code suivant :

```
class MyComponent extends mx.core.UIComponent;
```

Classe UIComponent (API)

Héritage MovieClip > Classe UIObject > UIComponent

Nom de classe Actionscript mx.core.UIComponent

Les méthodes, propriétés et événements de la classe UIComponent vous permettent de contrôler le comportement commun des composants visuels Flash.

Méthodes de la classe UIComponent

Le tableau suivant répertorie les méthodes de la classe UIComponent.

Méthode	Description
`UIComponent.getFocus()`	Renvoie une référence à l'objet ayant le focus.
`UIComponent.setFocus()`	Attribue le focus à l'occurrence de composant.

Méthodes héritées de la classe UIObject

Le tableau suivant répertorie les méthodes de la classe UIComponent héritées de la classe UIObject. Pour appeler ces méthodes à partir de l'objet UIComponent, utilisez le formulaire *`UIComponentInstance.methodName`*.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image active.
`UIObject.setSize()`	Redimensionne l'objet à la taille demandée.
`UIObject.setSkin()`	Définit une enveloppe dans l'objet.
`UIObject.setStyle()`	Définit la propriété de style sur l'objet ou la déclaration de style.

Propriétés de la classe UIComponent

Le tableau suivant répertorie les propriétés de la classe UIComponent.

Propriété	Description
`UIComponent.enabled`	Indique si le composant peut recevoir le focus et la saisie.
`UIComponent.tabIndex`	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Propriétés héritées de la classe UIObject

Le tableau suivant répertorie les propriétés de la classe UIComponent héritées de la classe UIObject. Pour accéder à ces propriétés à partir de l'objet UIComponent, utilisez le formulaire *`UIComponentInstance.propertyName`*.

Propriété	Description
`UIObject.bottom`	Position du bord inférieur de l'objet par rapport au bord inférieur de son parent. Lecture seule.
`UIObject.height`	Hauteur de l'objet, en pixels. Lecture seule.
`UIObject.left`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.right`	Position du bord droit de l'objet par rapport au bord droit de son parent. Lecture seule.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Position du bord supérieur de l'objet par rapport à son parent. Lecture seule.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Largeur de l'objet, en pixels. Lecture seule.
`UIObject.x`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.y`	Bord supérieur de l'objet, en pixels. Lecture seule.

Evénements de la classe UIComponent

Le tableau suivant répertorie les événements de la classe UIComponent.

Evénement	Description
`UIComponent.focusIn`	Diffusé lorsqu'un objet reçoit le focus.
`UIComponent.focusOut`	Diffusé lorsqu'un objet perd le focus.
`UIComponent.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIComponent.keyUp`	Diffusé lorsqu'une touche est relâchée.

Evénements hérités de la classe UIObject

Le tableau suivant répertorie les événements de la classe UIComponent hérités de la classe UIObject.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

UIComponent.enabled

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
componentInstance.enabled
```

Description

Propriété : indique si le composant peut accepter (`true`) ou non (`false`) le focus et les actions de la souris. La valeur par défaut est `true`.

Exemple

L'exemple suivant définit la propriété `enabled` d'un composant CheckBox sur `false` :

```
checkBoxInstance.enabled = false;
```

UIComponent.focusIn

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.focusIn = function(eventObj:Object) {
  //...
};
componentInstance.addEventListener("focusIn", listenerObject);
```

Utilisation 2 :

```
on (focusIn) {
  // ...
}
```

Description

Evénement : indique aux écouteurs que l'objet a reçu le focus clavier.

Le premier exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*componentInstance*) distribue un événement (dans ce cas, `focusIn`) qui est géré par une fonction (appelée aussi *gestionnaire*) sur un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque cet événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence de composant.

Exemple

Le code suivant désactive un composant Button, `btn`, lorsqu'un utilisateur tape dans le composant TextInput, `txt` et active le bouton lorsque l'utilisateur clique dessus :

```
var txt:mx.controls.TextInput;
var btn:mx.controls.Button;

var txtListener:Object = new Object();
txtListener.focusOut = function() {
  _root.btn.enabled = true;
}
txt.addEventListener("focusOut", txtListener);

var txtListener2:Object = new Object();
txtListener2.focusIn = function() {
  _root.btn.enabled = false;
}
txt.addEventListener("focusIn", txtListener2);
```

Voir aussi

`EventDispatcher.addEventListener()`, `UIComponent.focusOut`

UIComponent.focusOut

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
on(focusOut){
   ...
}
listenerObject = new Object();
listenerObject.focusOut = function(eventObject){
   ...
}
componentInstance.addEventListener("focusOut", listenerObject)
```

Description

Evénement : indique aux écouteurs que l'objet a perdu le focus clavier.

Le premier exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence de composant.

Le second exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`componentInstance`*) distribue un événement (dans ce cas, `focusOut`) qui est géré par une fonction (appelée aussi *gestionnaire*) sur un objet écouteur (*`listenerObject`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque cet événement est déclenché, il transmet automatiquement un objet événement (*`eventObject`*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

Le code suivant désactive un composant Button, `btn`, lorsqu'un utilisateur tape dans le composant TextInput, `txt` et active le bouton lorsque l'utilisateur clique dessus :

```
var txt:mx.controls.TextInput;
var btn:mx.controls.Button;

var txtListener:Object = new Object();
txtListener.focusOut = function() {
  _root.btn.enabled = true;
}
txt.addEventListener("focusOut", txtListener);

var txtListener2:Object = new Object();
txtListener2.focusIn = function() {
  _root.btn.enabled = false;
}
txt.addEventListener("focusIn", txtListener2);
```

Voir aussi

`EventDispatcher.addEventListener()`, `UIComponent.focusIn`

UIComponent.getFocus()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

componentInstance`.getFocus();`

Paramètres

Aucun.

Valeur renvoyée

Une référence à l'objet qui a actuellement le focus.

Description

Méthode : renvoie une référence à l'objet qui a le focus clavier.

Exemple

Le code suivant renvoie une référence à l'objet qui a le focus et l'affecte à la variable `tmp` :

```
var tmp = checkbox.getFocus();
```

UIComponent.keyDown

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
on(keyDown){
   ...
}
listenerObject = new Object();
listenerObject.keyDown = function(eventObject){
   ...
}
componentInstance.addEventListener("keyDown", listenerObject)
```

Description

Evénement : indique aux écouteurs lorsque l'utilisateur appuie sur une touche. Il s'agit d'un événement de très bas niveau que vous devez utiliser uniquement en cas de nécessité car il pourrait avoir une incidence sur les performances du système.

Le premier exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence de composant.

Le second exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`componentInstance`*) distribue un événement (dans ce cas, `keyDown`) qui est géré par une fonction (appelée aussi *gestionnaire*) sur un objet écouteur (*`listenerObject`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque cet événement est déclenché, il transmet automatiquement un objet événement (*`eventObject`*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

Le code suivant fait clignoter une icône lorsque l'utilisateur appuie sur une touche :

```
formListener.handleEvent = function(eventObj)
{
  form.icon.visible = !form.icon.visible;
}
form.addEventListener("keyDown", formListener);
```

UIComponent.keyUp

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
on(keyUp){
  ...
}
listenerObject = new Object();
listenerObject.keyUp = function(eventObject){
  ...
}
componentInstance.addEventListener("keyUp", listenerObject)
```

Description

Evénement : signale aux écouteurs que l'utilisateur a relâché une touche. Il s'agit d'un événement de bas niveau que vous devez utiliser uniquement en cas de nécessité car il pourrait avoir une incidence sur les performances du système.

Le premier exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence de composant.

Le second exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*componentInstance*) distribue un événement (dans ce cas, keyUp) qui est géré par une fonction (appelée aussi *gestionnaire*) sur un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque cet événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode EventDispatcher.addEventListener() sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

Le code suivant fait clignoter une icône lorsqu'une touche est relâchée :

```
formListener.handleEvent = function(eventObj)
{
  form.icon.visible = !form.icon.visible;
}
form.addEventListener("keyUp", formListener);
```

UIComponent.setFocus()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

componentInstance.setFocus();

Paramètres

Aucun.

Valeur renvoyée

Aucune.

Description

Méthode : définit le focus à cette occurrence de composant. L'occurrence avec le focus reçoit la saisie au clavier.

Exemple

Le code suivant donne le focus à l'occurrence `checkbox` :

```
checkbox.setFocus();
```

UIComponent.tabIndex

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`instance``.tabIndex`

Description

Propriété : nombre indiquant l'ordre de tabulation pour un composant dans un document.

Exemple

Le code suivant donne la valeur `tmp` à la propriété `tabIndex` de l'occurrence `checkbox` :

```
var tmp = checkbox.tabIndex;
```

CHAPITRE 53

Classe UIEventDispatcher

53

Nom de classe ActionScript mx.events.UIEventDispatcher

Héritage Classe EventDispatcher > UIEventDispatcher

La classe UIEventDispatcher est combinée à la classe UIComponent et permet aux composants d'émettre certains événements. Vous pouvez utiliser UIEventDispatcher pour qu'un objet qui n'hérite pas de l'interface UIComponent distribue certains événements.

REMARQUE	La classe UIEventDispatcher est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Méthodes de la classe UIEventDispatcher

Le tableau suivant présente la méthode de la classe UIEventDispatcher.

Méthode	Description
`UIEventDispatcher.removeEventListener()`	Supprime un écouteur enregistré d'une occurrence de composant. Cette méthode remplace la méthode `eventDispatcher.removeEventListenter()`.

Méthodes héritées de la classe EventDispatcher

Le tableau suivant répertorie les méthodes de la classe UIEventDispatcher héritées de la classe EventDispatcher. Pour appeler ces méthodes à partir de l'objet UIEventDispatcher, utilisez le formulaire *`UIEventDispatcherInstance.methodName`*.

Méthode	Description
`EventDispatcher.addEventListener()`	Enregistre un écouteur dans une occurrence de composant.
`EventDispatcher.dispatchEvent()`	Distribue un événement à tous les écouteurs enregistrés.

Evénements de la classe UIEventDispatcher

Le tableau suivant répertorie les événements de la classe UIEventDispatcher.

Méthode	Description
`UIEventDispatcher.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIEventDispatcher.keyUp`	Diffusé lorsqu'une touche est relâchée.
`UIEventDispatcher.load`	Diffusé lorsqu'un composant est chargé dans Flash Player.
`UIEventDispatcher.mouseDown`	Diffusé lorsque vous cliquez sur la souris.
`UIEventDispatcher.mouseOut`	Diffusé lorsque la souris se trouve en dehors d'une occurrence de composant.
`UIEventDispatcher.mouseOver`	Diffusé lorsque la souris est déplacée sur une occurrence de composant.
`UIEventDispatcher.mouseUp`	Diffusé lorsque vous cliquez sur la souris puis la relâchez.
`UIEventDispatcher.unload`	Diffusé lorsqu'un composant est déchargé de Flash Player.

UIEventDispatcher.keyDown

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
listenerObject = new Object();
listenerObject.keyDown = function(eventObject){
  // Insertion du code ici.
}
componentInstance.addEventListener("keyDown", listenerObject)
```

Description

Evénement ; diffusé à tous les écouteurs enregistrés lorsqu'une touche est enfoncée et que l'application Flash a le focus.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement.

UIEventDispatcher.keyUp

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
listenerObject = new Object();
listenerObject.keyUp = function(eventObject){
  // Insertion du code ici.
}
componentInstance.addEventListener("keyUp", listenerObject)
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsqu'une touche qui était enfoncée est relâchée et que l'application Flash a le focus.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement.

UIEventDispatcher.load

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
listenerObject = new Object();
listenerObject.load = function(eventObject){
  // Insertion du code ici.
}
componentInstance.addEventListener("load", listenerObject)
```

Description

Evénement ; diffusé à tous les écouteurs enregistrés lorsqu'un composant est chargé dans Flash Player.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement.

UIEventDispatcher.mouseDown

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
listenerObject = new Object();
listenerObject.mouseDown = function(eventObject){
  // Insertion du code ici.
}
componentInstance.addEventListener("mouseDown", listenerObject)
```

Description

Evénement ; diffusé à tous les écouteurs enregistrés lorsqu'une application Flash a le focus et que l'utilisateur clique sur la souris.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) au gestionnaire. Chaque objet événement est doté de propriétés contenant des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement.

UIEventDispatcher.mouseOut

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
listenerObject = new Object();
listenerObject.mouseOut = function(eventObject){
  // Insertion du code ici.
}
componentInstance.addEventListener("mouseOut", listenerObject)
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsqu'une application Flash a le focus et que la souris se trouve en dehors d'une occurrence de composant.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement.

UIEventDispatcher.mouseOver

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
listenerObject = new Object();
listenerObject.mouseOver = function(eventObject){
  // Insertion du code ici.
}
componentInstance.addEventListener("mouseOver", listenerObject)
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsqu'une application Flash a le focus et que la souris est déplacée sur une occurrence de composant.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement.

UIEventDispatcher.mouseUp

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
listenerObject = new Object();
listenerObject.mouseUp = function(eventObject){
  // Insertion du code ici.
}
componentInstance.addEventListener("mouseUp", listenerObject)
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsqu'une application Flash a le focus et que l'utilisateur clique sur la souris puis la relâche.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

UIEventDispatcher.removeEventListener()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004 ou Flash MX Professional 2004.

Utilisation

```
componentInstance.removeEventListener(event, listener)
```

Paramètres

event Chaîne portant le nom de l'événement.

listener Référence à une fonction ou à un objet écouteur.

Valeur renvoyée

Aucune.

Description

Méthode : annule l'enregistrement d'un objet d'écoute d'une occurrence de composant qui diffuse un événement. Cette méthode remplace l'événement `EventDispatcher.removeEventListener()` se trouvant dans la classe de base EventDispatcher.

UIEventDispatcher.unload

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
listenerObject = new Object();
listenerObject.unload = function(eventObject){
  // Insertion du code ici.
}
componentInstance.addEventListener("unload", listenerObject)
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsqu'un composant est déchargé de Flash Player.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) au gestionnaire. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement.

CHAPITRE 54

Classe UIObject

54

Héritage MovieClip > UIObject

Nom de classe ActionScript mx.core.UIObject

UIObject constitue la classe de base de tous les composants. Il ne s'agit pas d'un composant visuel.

REMARQUE

La classe UIObject est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

La classe UIObject enveloppe l'objet MovieClip ActionScript et contient des fonctions et des propriétés qui permettent aux composants de partager des comportements communs. L'habillage de la classe MovieClip permet à Adobe d'ajouter de nouveaux événements et, par la suite, d'étendre la fonctionnalité sans perte de contenu. L'habillage de la classe MovieClip permet également aux utilisateurs qui n'ont pas l'habitude des concepts d'animation et d'image propres à Flash d'utiliser des propriétés, des méthodes et des événements pour créer des applications basées sur les composants sans avoir besoin d'étudier ces concepts.

La classe UIObject implémente les éléments suivants :

- Styles
- Evénements
- Redimensionnement par mise à l'échelle

Pour utiliser les méthodes et propriétés de la classe UIObject, appelez-les directement du composant que vous utilisez. Par exemple, pour appeler la méthode `UIObject.setSize()` à partir du composant RadioButton, écrivez le code suivant :

```
myRadioButton.setSize(30, 30);
```

Vous devez seulement créer une occurrence de UIObject si vous créez un composant. Même dans ce cas, UIObject est souvent créée de façon implicite par d'autres sous-classes, telles que Button. Si vous n'avez pas besoin de créer une occurrence de classe UIObject, utilisez le code suivant :

```
class MyComponent extends UIObject;
```

Méthodes de la classe UIObject

Le tableau suivante répertorie les méthodes de la classe UIObject.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createLabel()`	Crée un sous-objet TextField à utiliser lors de la création de composants.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image active.
`UIObject.setSize()`	Redimensionne l'objet à la taille demandée.
`UIObject.setSkin()`	Définit une enveloppe dans l'objet.
`UIObject.setStyle()`	Définit la propriété de style sur l'objet ou la déclaration de style.

Propriétés de la classe UIObject

Le tableau suivant répertorie les propriétés de la classe UIObject.

Propriété	Description
`UIObject.bottom`	Position du bord inférieur de l'objet par rapport au bord inférieur de son parent. Lecture seule.
`UIObject.height`	Hauteur de l'objet, en pixels. Lecture seule.
`UIObject.left`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.right`	Position du bord droit de l'objet par rapport au bord droit de son parent. Lecture seule.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Position du bord supérieur de l'objet par rapport à son parent. Lecture seule.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Largeur de l'objet, en pixels. Lecture seule.
`UIObject.x`	Bord gauche de l'objet, en pixels. Lecture seule.
`UIObject.y`	Bord supérieur de l'objet, en pixels. Lecture seule.

Evénements de la classe UIObject

Le tableau suivant répertorie les événements de la classe UIObject.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

UIObject.bottom

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
componentInstance.bottom
```

Description

Propriété (lecture seule) ; nombre indiquant la position inférieure de l'objet, en pixels, par rapport à celle de son parent. Pour définir cette propriété, appelez `UIObject.move()`.

Exemple

Cet exemple permet de déplacer la case à cocher pour qu'elle soit alignée sous le bord inférieur de la zone de liste :

```
myCheckbox.move(myCheckbox.x, form.height - listbox.bottom);
```

UIObject.createClassObject()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
componentInstance.createClassObject(className, instanceName, depth,
  initObject)
```

Paramètres

`className` Objet indiquant la classe de la nouvelle occurrence.

`instanceName` Chaîne indiquant le nom de la nouvelle occurrence.

`depth` Nombre indiquant la profondeur de la nouvelle occurrence.

`initObject` Objet contenant des propriétés d'initialisation pour la nouvelle occurrence.

Valeur renvoyée

Un objet `UIObject` qui est une occurrence de la classe spécifiée.

Description

Méthode ; crée une occurrence d'un composant lors de l'exécution. Vous devez utiliser l'instruction `import` et spécifier le nom du package de classes avant d'appeler cette méthode. De plus, le composant doit se trouver dans la bibliothèque du fichier FLA.

Exemple

Le code suivant importe les actifs du composant Button, puis crée un sous-objet du composant Button :

```
import mx.controls.Button;
createClassObject(Button,"button2",5,{label:"Test Button"});
```

L'exemple suivant crée un objet `CheckBox` :

```
import mx.controls.CheckBox;
form.createClassObject(CheckBox, "cb", 0, {label:"Check this"});
```

Vous pouvez également spécifier le nom du package de classes en utilisant la syntaxe suivante :

```
createClassObject(mx.controls.Button, "button2", 5, {label:"Test Button"});
```

UIObject.createLabel()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
createLabel(name, depth, text)
```

Paramètres

`name` Chaîne du nom de l'occurrence.

`depth` Nombre indiquant la profondeur de la nouvelle occurrence.

`text` Texte de l'étiquette.

Valeur renvoyée

Un objet `TextField`.

Description

Méthode : crée un sous-objet `TextField`. Utilisée par la plupart des composants pour obtenir un objet de texte léger dans le composant tout en héritant des méthodes et des propriétés de dimension et de style du composant. Cette méthode est utilisée pour créer des composants. Le TextField créé est identique à l'objet `TextField` créé en appelant la méthode `MovieClip.createTextField()`,, mais il possède en outre les propriétés et les méthodes héritées de l'objet UIObject parent.

Un objet TextField qui utilise `UIObject.createLabel()` pour exécuter une création dans un composant peut tirer parti des méthodes UIObject suivantes héritées pour définir le dimensionnement et les styles dans le contexte de l'objet UIObject parent :

- TextField.getPreferredHeight() : nombre
- TextField.getPreferredWidth() : nombre
- TextField.setStyle(styleName : valeur de type chaîne)
- TextField.setSize(width : nombre, height : nombre)
- TextField.setValue(texte : chaîne) :

REMARQUE

L'objet TextFields créé avec `UIObject.createLabel()` possède une propriété initiale `TextField._visible` définie sur `false`. Cette propriété permet d'éviter l'oscillation qui peut survenir lorsque le composant parent appelle la méthode `UIObject.setSize()`. La propriété `TextField._visible` est définie sur true lorsque la méthode `UIObject.draw()`est appelée après le redimensionnement des objets enfant du composant parent.

Pour plus d'informations, reportez-vous à « Exemple simple de CellRenderer », à la page 114 pour consulter l'exemple de fichier MultilineCell.as.

Exemple

L'exemple suivant crée une occurrence TextField appelée `multiLineLabel` dans la méthode `UIComponent.createChildren()` d'un composant :

```
public function createChildren():Void {
  var myTextField_txt:TextField = this.createLabel("multiLineLabel", 900,
  "Hello World");
  // Définition de l'attribut de style fontSize de l'objet TextField.
  myTextField_txt.setStyle("fontSize", 18);
  // Définition de la dimension initiale de l'objet TextField.
  myTextField_txt.setSize(myTextField_txt.getPreferredWidth(),
  myTextField_txt.getPreferredHeight());
  // Définition de l'emplacement initial de l'objet TextField au centre
  // de la scène.
  myTextField_txt._x = (Stage.width/2) - (myTextField_txt._width/2);
  myTextField_txt._y = (Stage.height/2) - (myTextField_txt._height/2);
}
```

UIObject.createObject()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
componentInstance.createObject(linkageName, instanceName, depth,
  initObject)
```

Paramètres

linkageName Chaîne indiquant l'identifiant de liaison d'un symbole dans la bibliothèque.

instanceName Chaîne indiquant le nom de la nouvelle occurrence.

depth Nombre indiquant la profondeur de la nouvelle occurrence.

initObject Objet contenant des propriétés d'initialisation pour la nouvelle occurrence.

Valeur renvoyée

Un objet UIObject qui est une occurrence du symbole.

Description

Méthode : crée un sous-objet sur un objet. Cette méthode est généralement utilisée uniquement par des développeurs de composants ou des développeurs expérimentés.

Exemple

L'exemple suivant crée une occurrence CheckBox sur l'objet `form` :

```
form.createObject("CheckBox", "sym1", 0);
```

UIObject.destroyObject()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
destroyObject(instanceName)
```

Paramètres

instanceName Chaîne indiquant le nom d'occurrence de l'objet à détruire.

Valeur renvoyée

Aucune.

Description

Méthode : détruit une occurrence de composant.

Exemple

L'exemple suivant supprime l'occurrence TextInput `my_ti` lorsque vous cliquez sur le bouton. Un composant Button et un composant TextInput étant dans la bibliothèque du document en cours, ajoutez le code suivant sur la première image du scénario principal :

```
// Création d'occurrences TextInput et Button.
this.createClassObject(mx.controls.TextInput, "my_ti", 1, {text:"Hello
  World"});
this.createClassObject(mx.controls.Button, "my_button", 2, {label:"My
  Button"});
// Déplacement du bouton sous la saisie de texte.
my_button.move(my_ti.left, Stage.height - my_ti.bottom);

// Création d'un objet écouteur pour le clic de bouton.
var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object){
  destroyObject("my_ti");
}
// Ajout de l'écouteur.
my_button.addEventListener("click", buttonListener);
```

UIObject.doLater()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
componentInstance.doLater(target, "function")
```

Paramètres

target Référence à un scénario contenant la fonction spécifiée.

function Chaîne indiquant un nom de fonction à appeler après la transmission d'une image dans le clip de composant (ainsi les propriétés du composant définies dans l'inspecteur Propriétés ou l'inspecteur de composants sont disponibles).

Valeur renvoyée

Aucune.

Description

Méthode : appelle une fonction définie par l'utilisateur uniquement une fois que le composant a terminé de configurer toutes ses propriétés dans l'inspecteur Propriétés ou l'inspecteur des composants. Tous les composants qui héritent de UIObject ont la méthode `doLater()`.

Les propriétés des composants définies dans l'inspecteur Propriétés ou l'inspecteur de composants risquent de ne pas être disponibles immédiatement pour ActionScript dans le scénario. Par exemple, la tentative de suivre la propriété `label` d'un composant CheckBox en utilisant ActionScript sur la première image de votre fichier SWF échoue sans notification, même si le composant apparaît sur la scène comme prévu.

Même si les propriétés définies dans une classe ou un script d'image sont disponibles immédiatement, la plupart des propriétés qui sont affectées dans l'inspecteur Propriétés ou l'inspecteur des composants ne sont pas définies jusqu'à l'image suivante dans le composant lui-même.

Toute approche retardant l'accès à la propriété peut résoudre le problème, mais la solution la plus simple et la plus directe est d'utiliser la méthode `doLater()`.

Exemple

L'exemple suivant indique comment la méthode `doLater()` est utilisée :

```
// doLater() est appelée de l'occurrence du composant

myCheckBox.doLater(this, "delay");

// Fonction ou méthode appelée de doLater().

function delay() {
  trace(myCheckBox.label); // La propriété peut maintenant être suivie.
  // Toute instruction supplémentaire apparaît ici.
}
```

UIObject.draw

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.draw = function(eventObject:Object) {
  // ...
};
componentInstance.addEventListener("draw", listenerObject);
```

Utilisation 2 :

```
on (draw) {
  // ...
}
```

Description

Evénement : indique aux écouteurs que l'objet est sur le point de dessiner ses graphiques. Il s'agit d'un événement de bas niveau que vous devez utiliser uniquement en cas de nécessité, car il pourrait avoir une incidence sur les performances du système.

Le premier exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*componentInstance*) distribue un événement (dans ce cas, `draw`) qui est géré par une fonction (appelée aussi *gestionnaire*) sur un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence de composant.

Exemple

Le code suivant redessine l'objet `form2` lorsque l'objet `form` est dessiné :

```
formListener.draw = function(eventObj:Object) {
  form2.redraw(true);
}
form.addEventListener("draw", formListener);
```

Voir aussi

`EventDispatcher.addEventListener()`

UIObject.getStyle()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`componentInstance``.getStyle(`*`propertyName`*`)`

Paramètres

`propertyName` Chaîne indiquant le nom de la propriété de style (par exemple, « `fontWeight` », « `borderStyle` », etc.).

Valeur renvoyée

La valeur de la propriété de style. La valeur peut être de n'importe quel type de données.

Description

Méthode : obtient la propriété de style de l'objet ou de la déclaration de style. Si la propriété de style est un style d'héritage, les ancêtres de l'objet peuvent être la source de la valeur du style.

Pour obtenir une liste des styles pris en charge par chaque composant, consultez les entrées de composant individuelles. Reportez-vous également à « Utilisation de styles globaux, personnalisés et de classe dans le même document » dans *Utilisation des composants ActionScript 2.0*.

Exemple

Le code suivant définit la propriété de style `fontWeight` de l'occurrence `ib` sur bold si la propriété de style `fontWeight` de l'occurrence `cb` est bold :

```
if (cb.getStyle("fontWeight") == "bold") {
   ib.setStyle("fontWeight", "bold");
};
```

UIObject.height

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

componentInstance.height

Description

Propriété (lecture seule) : un nombre indiquant la hauteur de l'objet, en pixels. Pour modifier la propriété `height`, appelez `UIObject.setSize()`.

Exemple

L'exemple suivant augmente la hauteur de la case à cocher :

```
myCheckbox.setSize(myCheckbox.width, myCheckbox.height + 10);
```

UIObject.hide

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.hide = function(eventObject:Object) {
  // ...
};
componentInstance.addEventListener("hide", listenerObject);
```

Utilisation 2 :

```
on (hide) {
  // ...
}
```

Description

Evénement : diffusé lorsque la propriété `visible` de l'objet passe de `true` à `false`.

Exemple

Le gestionnaire suivant affiche un message dans le panneau de sortie lorsque l'objet auquel il est associé devient invisible.

```
on (hide) {
  trace("I've become invisible.");
}
```

Voir aussi

`UIObject.reveal`

UIObject.invalidate()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
componentInstance.invalidate()
```

Valeur renvoyée

Aucune.

Description

Méthode ; marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.

Cette méthode est surtout utile pour les développeurs de nouveaux composants personnalisés. Il est probable qu'un composant personnalisé prenne en charge un nombre d'opérations qui changent l'aspect du composant.

La meilleure façon de créer un composant est souvent de centraliser la logique pour mettre à jour l'aspect du composant dans la méthode `draw()`. Si le composant a une méthode `draw()`, vous pouvez appeler `invalidate()` sur le composant pour le redessiner. (Pour plus d'informations sur la définition d'une méthode `draw()`, reportez-vous à « Définition de la méthode draw() » dans *Utilisation des composants ActionScript 2.0*.)

Toutes les opérations qui changent l'aspect du composant peuvent appeler `invalidate()` au lieu de redessiner le composant elles-mêmes. Ceci présente des avantages : le code n'est pas dupliqué de façon inutile et plusieurs changements peuvent être combinés dans le nouveau dessin, au lieu d'en produire plusieurs redondants.

Exemple

L'exemple suivant marque `pBar` de l'occurrence ProgressBar pour que l'objet soit redessiné :

```
pBar.invalidate();
```

UIObject.left

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

componentInstance`.left`

Description

Propriété (lecture seule) : un nombre indiquant la longueur du bord gauche de l'objet en pixels par rapport à son parent. Pour définir cette propriété, appelez `UIObject.move()`.

UIObject.load

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.load = function(eventObject:Object) {
  // ...
};
componentInstance.addEventListener("load", listenerObject);
```

Utilisation 2 :

```
on (load) {
  //...
}
```

Description

Evénement : indique aux écouteurs que le sous-objet pour cet objet est en cours de création.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence de composant (*componentInstance*) distribue un événement (dans ce cas, `load`) qui est géré par une fonction (appelée aussi *gestionnaire*) dans un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée au déclenchement de l'événement. Lorsqu'il est déclenché, l'événement transmet automatiquement un objet événement (*eventObject*) à la méthode de l'objet écouteur. Chaque objet événement est doté de propriétés contenant des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence de composant.

Exemple

L'exemple suivant crée une occurrence de MySymbol une fois que l'occurrence `form` est chargée :

```
var formListener:Object = new Object();
formListener.load = function(eventObj:Object) {
  form.createObject("MySymbol", "sym1", 0);
};
form.addEventListener("load", formListener);
```

UIObject.move

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.move = function(eventObject:Object):Void {
  // ...
};
componentInstance.addEventListener("move", listenerObject);
```

Utilisation 2 :

```
on (move) {
  // ...
}
```

Description

Evénement : indique aux écouteurs que l'objet a été déplacé.

Le premier exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*componentInstance*) distribue un événement (dans ce cas, `move`) qui est géré par une fonction (appelée aussi *gestionnaire*) sur un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence de composant.

Exemple

L'exemple suivant appelle la méthode `move()` pour repositionner un composant Button, `my_button`, de sa position actuelle au coin supérieur gauche (10,10) de la scène :

```
var my_button:mx.controls.Button;
my_button.addEventListener("move", doMove);
function doMove(evt_obj:Object):Void {
  trace(evt_obj.target + " moved from {oldX:" + evt_obj.oldX + ", oldY:" +
  evt_obj.oldY + "} to {x:" + evt_obj.target.x + ", y:" + evt_obj.target.y
  + "}");
}
my_button.move(10, 10);
```

UIObject.move()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

componentInstance`.move(`*x*`, `*y*`, `*noEvent*`)`

Paramètres

x Nombre qui indique la position du coin supérieur gauche de l'objet par rapport à son parent.

y Nombre qui indique la position du coin supérieur droit de l'objet par rapport à son parent.

noEvent Valeur booléenne qui indique si l'événement move doit être distribué ou non.

Valeur renvoyée

Aucune.

Description

Méthode : déplace l'objet à l'emplacement demandé. Il est recommandé de transmettre uniquement des valeurs entières à `UIObject.move()` car sinon, le composant risque d'apparaître flou.

Exemple

L'exemple suivant permet de déplacer la case à cocher de 10 pixels vers la droite :

```
myCheckbox.move(myCheckbox.x + 10, myCheckbox.y);
```

L'exemple suivant appelle la méthode `move()` pour repositionner un composant Button, `my_button`, de sa position actuelle au coin supérieur gauche (10,10) de la scène :

```
var my_button:mx.controls.Button;
my_button.addEventListener("move", doMove);
function doMove(evt_obj:Object):Void {
  trace(evt_obj.target + " moved from {oldX:" + evt_obj.oldX + ", oldY:" +
  evt_obj.oldY + "} to {x:" + evt_obj.target.x + ", y:" + evt_obj.target.y
  + "}");
}
my_button.move(10, 10);
```

UIObject.redraw()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
componentInstance.redraw(always)
```

Paramètres

always Valeur booléenne. Si elle est définie sur `true`, la méthode dessine l'objet même si `invalidate()` n'a pas été appelé. Si elle est définie sur `false`, la méthode dessine l'objet uniquement si `invalidate()` a été appelé.

Valeur renvoyée

Aucune.

Description

Méthode : force la validation de l'objet pour qu'il soit dessiné dans l'image courante.

Exemple

L'exemple suivant crée une case à cocher et un bouton et les dessine car il n'est pas prévu que d'autres scripts modifient le formulaire :

```
form.createClassObject(mx.controls.CheckBox, "cb", 0);
form.createClassObject(mx.controls.Button, "b", 1);
form.redraw(true)
```

UIObject.resize

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.resize = function(eventObject:Object) {
  // ...
};
componentInstance.addEventListener("resize", listenerObject);
```

Utilisation 2 :

```
on (resize) {
  // ...
}
```

Description

Evénement : signale aux écouteurs qu'un objet a été redimensionné.

Le premier exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*componentInstance*) distribue un événement (dans ce cas, `resize`) qui est géré par une fonction (appelée aussi *gestionnaire*) sur un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. Chaque objet événement a des propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence de composant.

Exemple

L'exemple suivant appelle la méthode `setSize()` pour que `sym1` fasse la moitié de la largeur et un quart de la hauteur lorsque `form` est déplacé :

```
var formListener:Object = new Object();
formListener.resize = function(eventObj:Object):Void {
  form.sym1.setSize(sym1.width / 2, sym1.height / 4);
};
form.addEventListener("resize", formListener);
```

L'exemple suivant appelle la méthode `setSize()` pour redimensionner un composant Button, `my_button`, sur une largeur de 200 pixels et une hauteur de 100 pixels :

```
var my_button:mx.controls.Button;

my_button.addEventListener("resize", doSize);
function doSize(evt_obj:Object):Void {
  trace(evt_obj.target + " resized from {oldWidth:" + evt_obj.oldWidth + ",
  oldHeight:" + evt_obj.oldHeight + "} to {width:" + evt_obj.target.width +
  ", height:" + evt_obj.target.height + "}");
}
my_button.setSize(200, 100);
```

UIObject.reveal

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.reveal = function(eventObject:Object) {
  // ...
};
componentInstance.addEventListener("reveal", listenerObject);
```

Utilisation 2 :

```
on (reveal) {
  // ...
}
```

Description

Evénement : diffusé lorsque la propriété `visible` de l'objet passe de `false` à `true`.

Exemple

Le gestionnaire suivant affiche un message dans le panneau de sortie lorsque l'objet auquel il est associé devient visible.

```
on (reveal) {
  trace("I've become visible.");
}
```

Voir aussi

`UIObject.hide`

UIObject.right

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

componentInstance.right

Description

Propriété (lecture seule) ; un nombre indiquant le bord de l'objet sur la droite, en pixels, par rapport au bord droit de son parent. Pour définir cette propriété, appelez UIObject.move().

Exemple

L'exemple suivant permet de déplacer la case à cocher afin qu'elle s'aligne sous le bord droit de la zone de liste :

```
myCheckbox.move(form.width - listbox.right, myCheckbox.y);
```

UIObject.scaleX

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

componentInstance.scaleX

Description

Propriété : nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.

Exemple

L'exemple suivant double la largeur de la case à cocher et définit la variable `tmp` par rapport au facteur de redimensionnement horizontal :

```
checkbox.scaleX = 200;
var tmp:Number = checkbox.scaleX;
```

UIObject.scaleY

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

componentInstance.scaleY

Description

Propriété : nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.

Exemple

L'exemple suivant double la hauteur de la case à cocher et définit la variable `tmp` par rapport au facteur de redimensionnement vertical :

```
checkbox.scaleY = 200;
var tmp:Number = checkbox.scaleY;
```

UIObject.setSize()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

componentInstance.setSize(*width*, *height*, *noEvent*)

Paramètres

width Nombre indiquant la largeur de l'objet en pixels.

height Nombre indiquant la hauteur de l'objet en pixels.

noEvent Valeur booléenne qui indique si l'événement move doit être distribué ou non.

Valeur renvoyée

Aucune.

Description

Méthode : redimensionne l'objet à la taille requise. Il est recommandé de transmettre uniquement des valeurs entières à `UIObject.setSize()` car sinon, le composant risque d'apparaître flou. Cette méthode (ainsi que toutes les méthodes et propriétés de la classe UIObject) est disponible à partir de n'importe quelle occurrence de composant.

Lorsque vous appelez cette méthode sur une occurrence ComboBox, la liste déroulante est redimensionnée et la propriété `rowHeight` de la liste contenue est également modifiée.

REMARQUE	Certains composants permettent de modifier les dimensions de hauteur *ou* de largeur uniquement. Par exemple, les composants CheckBox et RadioButton ne permettent pas de modifier la hauteur.

Exemple

Cet exemple fixe la taille du composant `pBar` sur 100 pixels de largeur et 100 pixels de hauteur :

```
pBar.setSize(100, 100);
```

L'exemple suivant appelle la méthode `setSize()` pour redimensionner un composant Button, `my_button`, sur une largeur de 200 pixels et une hauteur de 100 pixels :

```
var my_button:mx.controls.Button;

my_button.addEventListener("resize", doSize);
function doSize(evt_obj:Object):Void {
  trace(evt_obj.target + " resized from {oldWidth:" + evt_obj.oldWidth + ",
  oldHeight:" + evt_obj.oldHeight + "} to {width:" + evt_obj.target.width +
  ", height:" + evt_obj.target.height + "}");
}
my_button.setSize(200, 100);
```

UIObject.setSkin()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
componentInstance.setSkin(id, linkageName)
```

Paramètres

id Nombre indiquant la profondeur de l'enveloppe dans le composant.

linkageName Chaîne indiquant un actif de la bibliothèque.

Valeur renvoyée

Une référence au clip (enveloppe) qui était associé.

Description

Méthode : définit une enveloppe dans l'occurrence de composant. Utilisez cette méthode dans un fichier de classe de composant lorsque vous créez un composant. Pour plus d'informations, consultez « Présentation de l'affectation d'enveloppes » dans *Utilisation des composants ActionScript 2.0*.

Vous ne pouvez pas utiliser cette méthode pour définir les enveloppes d'un composant lors de l'exécution (par exemple, la façon dont vous définissez les styles d'un composant lors de l'exécution).

Exemple

Cet exemple est une section de code issue du fichier de classe d'un nouveau composant appelé Shape. Il crée une variable, `themeShape` et la définit sur l'identifiant de liaison de l'enveloppe. Dans la méthode `createChildren()`, la méthode `setSkin()` est appelée et l'id `1` et la variable qui porte l'identifiant de liaison de l'enveloppe lui sont transmis :

```
class Shape extends UIComponent {

  static var symbolName:String = "Shape";
  static var symbolOwner:Object = Shape;
  var className:String = "Shape";

  var themeShape:String = "circle_skin"

  function Shape() {
  }

  function init(Void):Void {
    super.init();
  }

  function createChildren():Void {
    setSkin(1, themeShape);
    super.createChildren();
  }
}
```

UIObject.setStyle()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
componentInstance.setStyle(propertyName, value)
```

Paramètres

propertyName Chaîne indiquant le nom de la propriété de style. Les styles pris en charge dépendent du composant. Chaque composant possède un ensemble de styles différent que vous pouvez définir. Par exemple, « Personnalisation du composant TextArea », à la page 1229 affiche un tableau de styles, notamment `fontWeight`. Par conséquent, pour un composant TextArea, vous pouvez utiliser `fontWeight` comme paramètre *propertyName*.

value Valeur de la propriété. Si la valeur est une chaîne, elle doit être mise entre guillemets.

Valeur renvoyée

Aucune.

Description

Méthode : définit la propriété de style sur l'objet ou la déclaration de style. Si la propriété de style est un style d'héritage, les enfants de l'objet sont informés de la nouvelle valeur.

Pour obtenir la liste des styles pris en charge par chacun des composants, consultez leurs entrées respectives. Par exemple, les styles de composant Button sont répertoriés dans « Utilisation de styles avec le composant Button », à la page 95.

Pour améliorer les performances, vous pouvez modifier les styles avant de les charger, de les calculer et de les appliquer aux objets du fichier SWF. Si vous modifiez les styles avant de les charger et de les calculer, l'appel à `setStyle` devient superflu.

Adobe vous conseille de définir des propriétés sur chaque objet, car les objets sont instanciés pour améliorer les performances lorsque vous utilisez des styles. Lorsque vous liez de façon dynamique des occurrences à la scène, définissez des propriétés dans le paramètre `initObj` dans l'appel que vous effectuez vers `UIObject.createClassObject()`, comme dans le code ActionScript suivant :

```
createClassObject(ComponentClass, "myInstance", 0, {styleName:"myStyle",
  color:0x99CCFF});
```

REMARQUE

Cet exemple utilise la déclaration de style personnalisé myStyle custom. Pour changer plusieurs propriétés ou les propriétés de plusieurs occurrences de composant, créez une déclaration de style personnalisée. Flash rend un composant utilisant une déclaration de style personnalisée plus rapidement qu'un composant utilisant `UIObject.setStyle()` pour plusieurs propriétés. Pour plus d'informations, consultez « Définition de styles personnalisés pour des groupes de composants » dans *Utilisation des composants ActionScript 2.0*.

Exemple

Le code suivant définit la propriété de style `fontWeight` de l'occurrence de la case à cocher `cb` sur bold :

```
cb.setStyle("fontWeight", "bold");
```

UIObject.top

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`componentInstance``.top`

Description

Propriété (lecture seule) : un nombre indiquant le bord supérieur de l'objet en pixels par rapport à son parent. Pour définir cette propriété, appelez `UIObject.move()`.

UIObject.unload

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.unload = function(eventObject:Object):Void {
  // ...
};
componentInstance.addEventListener("unload", listenerObject);
```

Utilisation 2 :

```
on (unload) {
  // ...
}
```

Description

Evénement : indique aux écouteurs que les sous-objets de cet objet sont purgés.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence de composant (*componentInstance*) distribue un événement (dans ce cas, `unload`) qui est géré par une fonction (appelée aussi *gestionnaire*) dans un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée au déclenchement de l'événement. Lorsqu'il est déclenché, l'événement transmet automatiquement un objet événement (*eventObject*) à la méthode de l'objet écouteur. Chaque objet événement est doté de propriétés contenant des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence de composant.

Exemple

L'exemple suivant supprime `sym1` lorsque l'événement `unload` est déclenché :

```
function doUnload():Void {
  form.destroyObject(sym1);
}
form.addEventListener("unload", doUnload);
```

UIObject.visible

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

componentInstance.visible

Description

Propriété : valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).

Exemple

L'exemple suivant rend visible l'occurrence du composant Loader `myLoader` :

```
myLoader.visible = true;
```

UIObject.width

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

componentInstance.width

Description

Propriété (lecture seule) : un nombre indiquant la largeur de l'objet, en pixels. Pour changer la largeur, appelez `UIObject.setSize()`.

Exemple

L'exemple suivant rend la case à cocher plus large :

```
myCheckbox.setSize(myCheckbox.width + 10, myCheckbox.height);
```

UIObject.x

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

componentInstance.x

Description

Propriété (lecture seule) : un nombre indiquant le bord gauche de l'objet, en pixels. Pour définir cette propriété, appelez `UIObject.move()`.

Exemple

L'exemple suivant permet de déplacer la case à cocher de 10 pixels vers la droite :

```
myCheckbox.move(myCheckbox.x + 10, myCheckbox.y);
```

UIObject.y

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

componentInstance.y

Description

Propriété (lecture-seule) ; un nombre indiquant le bord supérieur de l'objet, en pixels. Pour définir cette propriété, appelez `UIObject.move()`.

Exemple

L'exemple suivant permet de déplacer la case à cocher de 10 pixels vers le bas :

```
myCheckbox.move(myCheckbox.x, myCheckbox.y + 10);
```

CHAPITRE 55

Composant UIScrollBar

55

Le composant UIScrollBar permet d'ajouter une barre de défilement à un champ de texte. Vous pouvez ajouter une barre de défilement à un champ de texte pendant la programmation ou lors de l'exécution avec ActionScript.

REMARQUE

Un composant UIScrollBar est pris en charge pour ActionScript 2.0 et ActionScript 3.0. Ce document présente le composant de la version 2. Si vous utilisez celui de la version 3, reportez-vous à « Utilisation du composant UIScrollBar » dans *Utilisation des composants ActionScript 3.0*.

Le composant UIScrollBar fonctionne comme toute autre barre de défilement. Il contient des boutons fléchés aux deux extrémités et un rail et un curseur de défilement entre les deux. Il peut être associé à n'importe quel bord d'un champ de texte et utilisé verticalement et horizontalement.

Utilisation du composant UIScrollBar

Pour utiliser le composant UIScrollBar, vérifiez que l'accrochage aux objets est activé (Affichage > Accrochage > Accrocher aux objets). Créez ensuite un champ de saisie de texte sur la scène et faites glisser le composant UIScrollBar du panneau Composants vers n'importe quel quadrant du cadre de sélection du champ de texte.

Si la longueur de la barre de défilement est inférieure à la taille combinée de ses flèches de défilement, elle ne s'affiche pas correctement. L'une des touches fléchées est masquée derrière l'autre. Flash ne fournit pas de détection des erreurs pour ceci. Dans ce cas, il peut être utile de masquer la barre de défilement avec ActionScript. Si la barre de défilement est dimensionnée de façon à ce qu'il y ait assez de place pour le curseur de défilement, Flash rend ce dernier invisible.

Contrairement à de nombreux autres composants, le composant UIScrollBar peut recevoir des actions de la souris en continu, comme lorsque l'utilisateur maintient le bouton de la souris enfoncé, plutôt que d'exiger des clics répétés.

Il n'existe pas d'interaction clavier avec le composant UIScrollBar.

Paramètres de UIScrollBar

Dans l'inspecteur des propriétés ou l'inspecteur de composants (option de menu Fenêtre > Inspecteur de composants), vous pouvez définir les paramètres de création suivants pour chaque occurrence UIScrollBar :

_targetInstanceName indique le nom de l'occurrence du champ de texte auquel le composant UIScrollBar est associé.

horizontal indique si la barre de défilement est orientée horizontalement (`true`) ou verticalement (`false`). La valeur par défaut est `false`.

Dans l'inspecteur de composants, vous pouvez définir les paramètres supplémentaires suivants pour chaque occurrence du composant UIScrollBar (Fenêtre > Inspecteur de composants) :

enabled est une valeur booléenne qui indique si le composant peut recevoir le focus et l'entrée. La valeur par défaut est `true`.

visible est une valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`). La valeur par défaut est `true`.

REMARQUE	Les propriétés minHeight et minWidth sont utilisées par des routines de dimensionnement internes. Elles sont définies dans UIObject et remplacées par différents composants selon vos besoins. Ces propriétés peuvent être utilisées pour créer un gestionnaire de dispositions personnalisées. Dans le cas contraire, la définition de ces propriétés dans l'inspecteur de composants n'a aucun effet visible.

Vous pouvez contrôler ces options et d'autres options d'un composant UIScrollBar à l'aide des propriétés, méthodes et événements d'ActionScript. Pour plus d'informations, voir « Classe UIScrollBar », à la page 1450.

Création d'une application avec le composant UIScrollBar

La procédure suivante explique comment ajouter un composant UIScrollBar à une application pendant la programmation.

Pour créer une application avec le composant UIScrollBar :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Créez un champ de texte dynamique et nommez l'occurrence, **myText** dans l'inspecteur des propriétés.
3. Dans l'inspecteur des propriétés, définissez le type de ligne du champ de saisie de texte sur Multiligne ou Multiligne sans retour si vous envisagez d'utiliser la barre de défilement horizontalement.
4. Dans l'image 1, utilisez ActionScript pour ajouter suffisamment de texte au champ de façon à ce que les utilisateurs doivent le faire défiler pour le visualiser dans son ensemble. Vous pouvez écrire le code suivant :

```
myText.text="When the moon is in the seventh house and Jupiter aligns
  with Mars, then peace will guide the planet and love will rule the
  stars."
```

REMARQUE	Vérifiez que le champ de texte sur la scène est assez petit pour que vous deviez le faire défiler afin de visualiser tout le texte. Si ce n'est pas le cas, la barre de défilement ne s'affiche pas ou risque d'apparaître sur deux lignes sans poignée miniature (partie que vous faites glisser pour faire défiler le contenu).

5. Vérifiez que l'accrochage aux objets est activée (Affichage > Accrochage > Accrocher aux objets).
6. Faites glisser une occurrence de UIScrollBar du panneau Composants sur le champ de saisie de texte près du côté auquel vous souhaitez l'associer. Le champ de texte et le composant doivent se chevaucher lorsque vous relâchez la souris de façon à ce que le composant soit correctement lié au champ.

   La propriété `_targetInstanceName` du composant est renseignée automatiquement avec le nom de l'occurrence du champ de texte dans les inspecteurs des propriétés et de composants. Si elle n'apparaît pas dans l'onglet Paramètres, vous n'avez peut-être pas suffisamment recouvert l'occurrence UIScrollBar.
7. Sélectionnez Contrôle > Tester l'animation.

   L'application est exécutée et la barre de défilement fait défiler le contenu du champ de texte.

Vous pouvez également créer une occurrence de composant UIScrollBar et l'associer à un champ de texte lors de l'exécution avec ActionScript.

Le code suivant crée une occurrence du composant UIScrollBar orientée verticalement, l'associe au côté droit d'une occurrence de champ de texte appelée **my_txt** et définit la taille de la barre de défilement en fonction de la taille du champ de texte :

```
/**
 Requiert :
  - Composant UIScrollBar dans la bibliothèque
*/
// Création d'un champ de texte.
this.createTextField("my_txt", 10, 10, 20, 200, 100);
my_txt.wordWrap = true;

this.createClassObject(mx.controls.UIScrollBar, "my_sb", 20);

// Définition du champ de texte cible pour la barre de défilement.
my_sb.setScrollTarget(my_txt);

// Dimensionnement en fonction du champ de texte.
my_sb.setSize(16, my_txt._height);

// Déplacement près du champ de texte.
my_sb.move(my_txt._x + my_txt._width, my_txt._y);

// Chargement du texte à afficher et définition du gestionnaire onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_txt.text = src;
    } else {
        my_txt.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

Personnalisation du composant UIScrollBar

Vous pouvez transformer un composant UIScrollBar de façon horizontale et verticale pendant la programmation et lors de l'exécution. Néanmoins, un composant UIScrollBar vertical ne vous permet pas de modifier la largeur, et un composant UIScrollBar horizontal ne vous permet pas de modifier la hauteur. Lors de la programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. Lors de l'exécution, utilisez la méthode `setSize()` (reportez-vous à `UIObject.setSize()`) ou toute propriété et méthode applicable de la classe UIScrollBar.

REMARQUE

Si vous utilisez la méthode `UIObject.setSize()`, vous pouvez uniquement modifier la hauteur ou la largeur de l'occurrence selon que cette dernière est une barre de défilement verticale ou horizontale. Par conséquent, la méthode `setSize()` ignore les paramètres `height` ou `width`.

Notez, cependant, qu'avec le thème Halo, la largeur d'une barre de défilement orientée verticalement et la hauteur d'une barre de défilement orientée horizontalement doivent être de 16 pixels. Ces dimensions sont déterminées strictement par le thème courant utilisé avec la barre de défilement. Seule la dimension d'une barre de défilement qui correspond à sa longueur peut être modifiée.

Vous pouvez personnaliser l'aspect d'une occurrence de UIScrollBar en utilisant des styles et des enveloppes.

Utilisation de styles avec le composant UIScrollBar

Le composant UIScrollBar prend en charge les styles suivants :

Style	Thème	Description
`themeColor`	Halo	Couleur de base d'un composant. `"haloGreen"`, `"haloBlue"` et `"haloOrange"` sont des valeurs possibles. La valeur par défaut est `"haloGreen"`.
`scrollTrackColor`	Sample	Couleur d'arrière-plan du rail de défilement. La valeur par défaut est 0xCCCCCC (gris clair).
`symbolColor`	Sample	Couleur des flèches de défilement Haut et Bas. La valeur par défaut est 0x000000 (noir).
`symbolDisabledColor`	Sample	Couleur des flèches de défilement Haut et Bas dans une barre de défilement désactivée. La valeur par défaut est 0x848384 (gris foncé).

Utilisation d'enveloppes avec le composant UIScrollBar

Le composant UIScrollBar utilise 13 enveloppes pour le rail, le curseur de défilement et les boutons. Pour personnaliser ces éléments d'enveloppe, modifiez les symboles dans le dossier Flash UI Components 2/Themes/MMDefault/ScrollBar Assets/States. Pour plus d'informations, consultez « A propos de l'application d'enveloppes aux composants » dans *Utilisation des composants ActionScript 2.0*.

Les barres de défilement horizontale et verticale utilisent les mêmes enveloppes verticales. Lors de l'affichage d'une barre de défilement horizontale, le composant UIScrollBar fait pivoter les enveloppes comme nécessaire.

Le composant UIScrollBar prend en charge les propriétés d'enveloppe suivantes :

Propriété	Description
`upArrowUpName`	Etat relevé (normal) des boutons haut et gauche. La valeur par défaut est `ScrollUpArrowUp`.
`upArrowOverName`	Etat survolé des boutons haut et gauche. La valeur par défaut est `ScrollUpArrowOver`.
`upArrowDownName`	Etat enfoncé des boutons haut et gauche. La valeur par défaut est `ScrollUpArrowDown`.
`downArrowUpName`	Etat relevé (normal) des boutons bas et droit. La valeur par défaut est `ScrollDownArrowUp`.
`downArrowOverName`	Etat survolé des boutons bas et droit. La valeur par défaut est `ScrollDownArrowOver`.
`downArrowDownName`	Etat enfoncé des boutons bas et droit. La valeur par défaut est `ScrollDownArrowDown`.
`scrollTrackName`	Symbole utilisé pour le rail (arrière-plan) de la barre de défilement. La valeur par défaut est `ScrollTrack`.
`scrollTrackOverName`	Symbole utilisé pour le rail de défilement (arrière-plan) lorsqu'il est survolé. La valeur par défaut est `undefined`.
`scrollTrackDownName`	Symbole utilisé pour le rail de défilement (arrière-plan) lorsqu'il est enfoncé. La valeur par défaut est `undefined`.
`thumbTopName`	Embouts haut et gauche du curseur de défilement. La valeur par défaut est `ScrollThumbTopUp`.
`thumbMiddleName`	Partie centrale (extensible) du curseur. La valeur par défaut est `ScrollThumbMiddleUp`.

Propriété	Description
`thumbBottomName`	Embouts bas et droit du curseur. La valeur par défaut est `ScrollThumbBottomUp`.
`thumbGripName`	Poignée affichée devant le curseur. La valeur par défaut est `ScrollThumbGripUp`.

L'exemple suivant démontre comment placer une ligne fine vide au milieu du rail de défilement :

Pour créer des symboles de clip pour les enveloppes de UIScrollBar :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Choisissez Fichier > Importer > Ouvrir une bibliothèque externe et sélectionnez le fichier HaloTheme.fla.

   Ce fichier est stocké dans le dossier Configuration au niveau de l'application. Pour connaître son emplacement exact dans votre système d'exploitation, reportez-vous à « Présentation des thèmes » dans *Utilisation des composants ActionScript 2.0*.
3. Dans le panneau Bibliothèque du thème, développez Flash UI Components 2/Themes/MMDefault et faites glisser le dossier ScrollBar Assets vers la bibliothèque de votre document.
4. Développez le dossier ScrollBar Assets/States dans la bibliothèque de votre document.
5. Ouvrez les symboles que vous souhaitez personnaliser pour la modification.

   Par exemple, ouvrez le symbole ScrollTrack.
6. Personnalisez le symbole selon vos besoins.

   Par exemple, dessinez un rectangle noir au milieu du rail à l'aide d'un rectangle 1 x 4 aux coordonnées (8,0).
7. Répétez les étapes 5 et 6 pour tous les symboles devant être personnalisés.

   Par exemple, dessinez la même ligne sur le symbole ScrollTrackDisabled.
8. Cliquez sur le bouton Précédent pour revenir au scénario principal.
9. Créez une occurrence de TextField de type input sur la scène.
10. Faites glisser un composant UIScrollBar vers l'occurrence de TextField.
11. Sélectionnez Contrôle > Tester l'animation.

Classe UIScrollBar

Héritage MovieClip > Classe UIObject > Classe UIComponent > ScrollBar > UIScrollBar

Nom de classe ActionScript mx.controls.UIScrollBar

Les propriétés de la classe UIScrollBar vous permettent de régler la position et le volume de défilement lorsque l'utilisateur clique sur les flèches ou le rail de défilement.

Contrairement à la plupart des autres composants, les événements sont diffusés lorsque le bouton de la souris est enfoncé et continuent à diffuser jusqu'à ce que l'utilisateur relâche le bouton de la souris.

Toutes les classes de composants ont une propriété `version` qui correspond à une propriété de classe. Les propriétés de classe ne sont disponibles que pour la classe elle-même. La propriété `version` renvoie une chaîne qui indique la version du composant. Pour accéder à cette propriété, utilisez le code suivant :

```
trace(mx.controls.UIScrollBar.version);
```

REMARQUE

Le code `trace(myUIScrollBarInstance.version);` renvoie `undefined`.

Méthodes de la classe UIScrollBar

Le tableau suivant présente la méthode de la classe UIScrollBar.

Méthode	Description
`UIScrollBar.setScrollProperties()`	Définit la plage de défilement de la barre de défilement ainsi que la taille du champ de texte à laquelle la barre de défilement est associée.
`UIScrollBar.setScrollTarget()`	Affecte la barre de défilement à un champ de texte.

Méthodes héritées de la classe UIObject

Le tableau suivant répertorie les méthodes de la classe UIScrollBar héritées de la classe UIObject. Pour appeler ces méthodes à partir de l'objet UIScrollBar, utilisez le formulaire *UIScrollBarInstance.methodName*.

Méthode	Description
UIObject.createClassObject()	Crée un objet dans la classe spécifiée.
UIObject.createObject()	Crée un sous-objet dans un objet.
UIObject.destroyObject()	Détruit une occurrence de composant.
UIObject.doLater()	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
UIObject.getStyle()	Obtient la propriété de style de l'objet ou de la déclaration de style.
UIObject.invalidate()	Marque l'objet pour qu'il soit redessiné dans le prochain intervalle d'image.
UIObject.move()	Déplace l'objet à l'emplacement demandé.
UIObject.redraw()	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
UIObject.setSize()	Redimensionne l'objet à la taille demandée.
UIObject.setSkin()	Définit une enveloppe dans l'objet.
UIObject.setStyle()	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant répertorie les méthodes de la classe UIScrollBar héritées de la classe UIComponent. Pour appeler ces méthodes à partir de l'objet UIScrollBar, utilisez le formulaire *UIScrollBarInstance.methodName*.

Méthode	Description
UIComponent.getFocus()	Renvoie une référence à l'objet ayant le focus.
UIComponent.setFocus()	Attribue le focus à l'occurrence de composant.

Propriétés de la classe UIScrollBar

Le tableau suivant répertorie les propriétés de la classe UIScrollBar.

Propriété	Description
`UIScrollBar.lineScrollSize`	Nombre de lignes ou de pixels qui défilent lorsque l'utilisateur clique sur les boutons fléchés de la barre de défilement.
`UIScrollBar.pageScrollSize`	Nombre de lignes ou de pixels qui défilent lorsque l'utilisateur clique sur le rail de la barre de défilement.
`UIScrollBar.scrollPosition`	Position de défilement courante de la barre de défilement.
`UIScrollBar._targetInstanceName`	Nom d'instance du champ de texte associée à l'occurrence de UIScrollBar.
`UIScrollBar.horizontal`	Valeur booléenne indiquant si la barre de défilement est orientée à la verticale (`false`), par défaut, ou à l'horizontale (`true`).

Propriétés héritées de la classe UIObject

Le tableau suivant répertorie les propriétés de la classe UIScrollBar héritées de la classe UIObject. Pour accéder à ces propriétés à partir de l'objet UIScrollBar, utilisez le formulaire *`UIScrollBarInstance.propertyName`*.

Propriété	Description
`UIObject.bottom`	Lecture seule ; position du bord inférieur de l'objet par rapport au bord inférieur de son parent.
`UIObject.height`	Lecture seule ; hauteur de l'objet, en pixels.
`UIObject.left`	Lecture seule ; bord gauche de l'objet, en pixels.
`UIObject.right`	Lecture seule ; position du bord droit de l'objet par rapport au bord droit de son parent.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Lecture seule ; renvoie la position du bord supérieur de l'objet par rapport à son parent.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).

Propriété	Description
`UIObject.width`	Lecture seule ; largeur de l'objet, en pixels.
`UIObject.x`	Lecture seule ; bord gauche de l'objet, en pixels.
`UIObject.y`	Lecture seule ; bord supérieur de l'objet, en pixels.

Propriétés héritées de la classe UIComponent

Le tableau suivant répertorie les propriétés de la classe UIScrollBar héritées de la classe UIComponent. Pour accéder à ces propriétés à partir de l'objet UIScrollBar, utilisez le formulaire *`UIScrollBarInstance.propertyName`*.

Propriété	Description
`UIComponent.enabled`	Indique si le composant peut recevoir le focus et la saisie.
`UIComponent.tabIndex`	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Evénements de la classe UIScrollBar

Le tableau suivant présente l'événement de la classe UIScrollBar.

Evénement	Description
`UIScrollBar.scroll`	Diffusé lorsque vous cliquez n'importe où sur la barre de défilement.

Evénements hérités de la classe UIObject

Le tableau suivant répertorie les événements de la classe UIScrollBar hérités de la classe UIObject.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.

Evénement	Description
UIObject.reveal	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
UIObject.unload	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant répertorie les événements de la classe UIScrollBar hérités de la classe UIComponent.

Evénement	Description
UIComponent.focusIn	Diffusé lorsqu'un objet reçoit le focus.
UIComponent.focusOut	Diffusé lorsqu'un objet perd le focus.
UIComponent.keyDown	Diffusé lorsqu'une touche est enfoncée.
UIComponent.keyUp	Diffusé lorsqu'une touche est relâchée.

UIScrollBar.horizontal

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

scrollBarInstance.horizontal

Description

Propriété ; indique si la barre de défilement est orientée verticalement (false) ou horizontalement (true).

Cette propriété peut être testée et définie. La valeur par défaut est false.

Exemple

L'exemple suivant utilise la propriété `horizontal` pour définir la barre de défilement appelée `my_sb` sur une orientation horizontale, puis affiche le texte dans le composant TextField `my_txt` :

```
/**
 Requiert :
  - Composant UIScrollBar dans la bibliothèque
*/
// Création du champ de texte.
this.createTextField("my_txt", 10, 10, 20, 200, 100);
my_txt.wordWrap = false;

my_txt.text = "Mary had a little lamb whose fleece " +
"was white as snow and everywhere that Mary went the " +
"lamb was sure to go."

// Création d'une barre de défilement.
this.createClassObject(mx.controls.UIScrollBar, "my_sb", 20);
my_sb.horizontal = true;

// Définition du champ de texte cible pour la barre de défilement.
my_sb.setScrollTarget(my_txt);
// Dimensionnement en fonction du champ de texte.
my_sb.setSize(my_txt._width, 16);

// Déplacement vers la base du champ de texte.
my_sb.move(my_txt._x, my_txt._y + my_txt._height);
```

UIScrollBar.lineScrollSize

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

scrollBarInstance.lineScrollSize

Description

Propriété : obtient ou fixe le nombre de lignes ou de pixels qui défilent lorsque l'utilisateur clique sur les boutons fléchés du composant UIScrollBar. Si la barre de défilement est orientée verticalement, la valeur est un nombre de lignes. Si la barre de défilement est orientée horizontalement, la valeur est un nombre de pixels.

La valeur par défaut est 1.

Exemple

L'exemple suivant crée une barre de défilement pour faire défiler, dans un champ de texte, du texte qu'il charge depuis une page Web. L'exemple définit la propriété `lineScrollSize` pour faire défiler deux lignes à chaque clic d'un bouton fléché :

```
/**
 Requiert :
  - Composant UIScrollBar dans la bibliothèque
*/

this.createTextField("my_txt", 10, 10, 20, 200, 100);
my_txt.wordWrap = true;

this.createClassObject(mx.controls.UIScrollBar, "my_sb", 20);

// Définition du champ de texte cible.
my_sb.setScrollTarget(my_txt);

// Dimensionnement en fonction du champ de texte.
my_sb.setSize(16, my_txt._height);

// Déplacement près du champ de texte.
my_sb.move(my_txt._x + my_txt._width, my_txt._y);

// Défilement de 2 lignes par clic sur une flèche de défilement.
my_sb.lineScrollSize = 2;

// Défilement de 5 lignes par clic sur le rail de défilement.
my_sb.pageScrollSize = 5;

// Chargement du texte à afficher et définition du gestionnaire onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_txt.text = src;
    } else {
        my_txt.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

UIScrollBar.pageScrollSize

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

scrollBarInstance.pageScrollSize

Description

Propriété : obtient ou fixe le nombre de lignes ou de pixels qui défilent lorsque l'utilisateur clique sur le rail du composant UIScrollBar. Si la barre de défilement est orientée verticalement, la valeur est un nombre de lignes. Si la barre de défilement est orientée horizontalement, la valeur est un nombre de pixels.

Vous pouvez également définir cette valeur en transmettant un paramètre *pageSize* avec la méthode UIScrollBar.setScrollTarget().

Exemple

L'exemple suivant crée une barre de défilement pour faire défiler, dans un champ de texte, du texte qu'il charge depuis une page Web. L'exemple définit la propriété pageScrollSize pour faire défiler cinq lignes de texte chaque fois que l'utilisateur clique sur le rail de défilement :

```
/**
 Requiert :
  - Composant UIScrollBar dans la bibliothèque
*/

this.createTextField("my_txt", 10, 10, 20, 200, 100);
my_txt.wordWrap = true;

this.createClassObject(mx.controls.UIScrollBar, "my_sb", 20);

// Définition du champ de texte cible.
my_sb.setScrollTarget(my_txt);

// Dimensionnement en fonction du champ de texte.
my_sb.setSize(16, my_txt._height);

// Déplacement près du champ de texte.
my_sb.move(my_txt._x + my_txt._width, my_txt._y);
```

```
// Défilement de 2 lignes par clic d'une flèche de défilement.
my_sb.lineScrollSize = 2;

// Défilement de 5 lignes par clic d'un rail de défilement.
my_sb.pageScrollSize = 5;

// Chargement du texte à afficher et définition du gestionnaire onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_txt.text = src;
    } else {
        my_txt.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

UIScrollBar.scroll

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.scroll = function(eventObject:Object) {
  // ...
};
scrollBarInstance.addEventListener("scroll", listenerObject)
```

Utilisation 2 :

```
on (scroll) {
  // ...
}
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque l'utilisateur clique sur la barre de défilement (puis relâche le bouton de la souris). La propriété `UIScrollBar.scrollPosition` et l'image à l'écran de la barre de défilement sont mises à jour avant la diffusion de cet événement.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement dans lequel le script est placé sur une image dans le scénario qui contient l'occurrence de composant. Une occurrence de composant (*`scrollBarInstance`*) distribue un événement (ici, `scroll`) géré par une fonction (également appelée *gestionnaire*) associée à un objet écouteur (*`listenerObject`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute. La méthode est appelée au déclenchement de l'événement. Lorsque il est déclenché, l'événement transmet automatiquement un objet événement (*`eventObject`*) à la méthode d'objet écouteur. L'objet événement est doté de propriétés qui contiennent des informations sur l'événement.
Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `addEventListener()` (voir `EventDispatcher.addEventListener()`) sur l'occurrence du composant qui diffuse l'événement afin d'enregistrer l'écouteur avec cette occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Outre les propriétés normales de l'objet événement (`type` et `target`), l'objet événement pour l'événement `scroll` inclut une troisième propriété appelée `direction`. La propriété `direction` contient une chaîne décrivant la façon dont la barre de défilement est orientée. Les valeurs possibles de la propriété `direction` sont `vertical` (valeur par défaut) et `horizontal`.

Pour plus d'informations sur les propriétés de l'objet événement `type` et `target`, reportez-vous à « Objets événement », à la page 516.

Le second exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence du composant UIScrollBar. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence du composant. Par exemple, le code suivant, joint à l'occurrence du composant UIScrollBar `myUIScrollBarComponent`, envoie « _level0.myUIScrollBarComponent » dans le panneau Sortie :

```
on (scroll) {
  trace(this);
}
```

Exemple

L'exemple suivant implémente l'utilisation 1 et crée un objet écouteur appelé `sbListener` avec un gestionnaire d'événements `scroll` :

```
/**
 Requiert :
  - Composant UIScrollBar dans la bibliothèque
*/

this.createTextField("my_txt", 10, 10, 20, 200, 100);
my_txt.wordWrap = true;

this.createClassObject(mx.controls.UIScrollBar, "my_sb", 20);

// Définition du champ de texte cible.
my_sb.setScrollTarget(my_txt);

// Dimensionnement en fonction du champ de texte.
my_sb.setSize(16, my_txt._height);

// Déplacement près du champ de texte.
my_sb.move(my_txt._x + my_txt._width, my_txt._y);

// Création d'un objet écouteur.
var sbListener:Object = new Object();
sbListener.scroll = function(evt_obj:Object){
   // Insertion du code afin de gérer l'événement « scroll ».
   trace("text is scrolling");
}
// Ajout de l'écouteur.
my_sb.addEventListener("scroll", sbListener);

// Chargement du texte à afficher et définition du gestionnaire onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_txt.text = src;
    } else {
        my_txt.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

Le code suivant implémente l'utilisation 2. Le code est associé à l'occurrence du composant UIScrollBar et envoie un message au panneau Sortie lorsque l'utilisateur clique sur la barre de défilement. Le gestionnaire `on()` doit être directement lié à l'occurrence UIScrollBar.

```
on (scroll) {
  trace("UIScrollBar component was clicked");
}
```

UIScrollBar.scrollPosition

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

scrollBarInstance.scrollPosition

Description

Propriété ; obtient ou définit la position de défilement actuelle de la case de défilement (miniature) lorsqu'une nouvelle valeur `scrollPosition` est définie. La valeur `scrollPosition` dépend de l'utilisation ou non de l'occurrence UIScrollBar pour le défilement vertical ou horizontal.

Définissez le défilement de l'occurrence cible de la barre de défilement séparément en utilisant la syntaxe suivante :

```
my_scrollbar._targetInstanceName.scroll = 20;
```

Si l'occurrence UIScrollBar est utilisée pour le défilement vertical (l'utilisation la plus courante), la valeur de `scrollPosition` est un nombre entier avec une plage qui commence par 0 et se termine par un nombre égal au nombre total de lignes dans le champ de texte divisé par le nombre de lignes pouvant être affichées simultanément dans ce champ de texte. Si la valeur `scrollPosition` est définie sur un nombre supérieur à cette plage, le champ de texte fait simplement défiler le texte jusqu'à la fin.

Pour définir la case de défilement (miniature) sur la première ligne, définissez `scrollPosition` sur 0.

Pour définir la case de défilement (miniature) sur la fin, définissez la valeur `scrollPosition` sur le nombre de lignes de texte dans le champ de texte moins 1. Vous pouvez déterminer le nombre de lignes en récupérant la valeur de la propriété `maxscroll` du champ de texte.

Si l'occurrence UIScrollBar est utilisée pour le défilement horizontal, la valeur `scrollPosition` est un nombre entier compris entre 0 et la largeur du champ de texte, en pixels. Vous pouvez déterminer la largeur du champ de texte en pixels en récupérant la valeur de la propriété `maxhscroll` du champ de texte.

La valeur par défaut de `scrollPosition` est 0.

Exemple

L'exemple suivant définit la position du texte 20 :

```
/**
 Requiert :
  - Composants UIScrollBar et Button dans la bibliothèque
*/
this.createTextField("my_txt", 10, 10, 20, 200, 100);
this.createClassObject(mx.controls.UIScrollBar, "my_sb", 20);
this.createClassObject(mx.controls.Button, "my_bt", 30, {label: "Scroll"});

my_txt.wordWrap = true;
my_bt.move(300, 100);

// Définition du champ de texte cible.
my_sb.setScrollTarget(my_txt);

// Déplacement près du champ de texte.
my_sb.move(my_txt._x + my_txt._width, my_txt._y);

// Chargement du texte à afficher et définition du gestionnaire onData.
var my_lv:LoadVars = new LoadVars();

my_lv.onData = function(src:String) {
if (src != undefined) {
  my_txt.text = src;
  } else {
    my_txt.text = "Error loading text.";

  }
};

my_lv.load("http://www.helpexamples.com/flash/lorem.txt");

var scroll_listener = new Object();
scroll_listener.click = function() {
  my_sb.scrollPosition = 20;
  my_txt.scroll = 20;
};
my_bt.addEventListener("click", scroll_listener);
```

UIScrollBar.setScrollProperties()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

scrollBarInstance.setScrollProperties(*pageSize*, *minPos*, *maxPos*)

Paramètres

pageSize Nombre d'éléments pouvant être affichés dans la zone d'affichage. Ce paramètre définit la taille du cadre de sélection du champ de texte. Si la barre de défilement est verticale, cette valeur est un nombre de lignes de texte ; si elle est horizontale, un nombre de pixels.

minPos Ce paramètre fait référence à la ligne de texte ayant le numéro le plus petit lorsque la barre de défilement est utilisée verticalement ou le pixel ayant le numéro le plus petit dans le cadre de sélection du champ de texte lorsqu'elle est utilisée horizontalement. La valeur est généralement égale à 0.

maxPos Cette valeur fait référence à la ligne de texte ayant le numéro le plus grand lorsque la barre de défilement est utilisée verticalement ou le pixel ayant le numéro le plus grand dans le cadre de sélection du champ de texte lorsqu'elle est utilisée horizontalement.

Description

Méthode : définit la plage de défilement de la barre de défilement ainsi que la taille du champ de texte à laquelle la barre de défilement est associée. Cette méthode est principalement utilisée lorsque vous liez un composant UIScrollBar à un champ de texte à l'exécution (à l'aide de `UIScrollBar.setScrollTarget()`) plutôt qu'à la création, et l'affectation ne provoque pas la diffusion des événements change par le champ de texte. Si vous utilisez la méthode `replaceText` pour définir le texte du champ de texte, vous devez utiliser `setScrollProperties()` pour mettre à jour les barres de défilement.

Les valeurs `minPos` et `maxPos` sont utilisées conjointement par le composant UIScrollBar pour déterminer la plage de défilement pour la barre de défilement et le champ de texte associé.

Exemple

L'exemple suivant configure un composant UIScrollBar de façon à afficher 10 lignes de texte à la fois dans le champ de texte en dehors d'une plage comprise entre 0 et 99 lignes :

```
my_sb.setScrollProperties(10, 0, 99);
```

UIScrollBar.setScrollTarget()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

scrollBarInstance.setScrollTarget(*textInstance*)

Paramètres

textInstance Champ de texte à affecter à la barre de défilement.

Description

Méthode : attribue un composant UIScrollBar à une occurrence de champ de texte.
Si vous devez associer un champ de texte et un composant UIScrollBar lors de l'exécution, utilisez cette méthode.

Exemple

L'exemple suivant crée une barre de défilement pour faire défiler, dans un champ de texte, du texte qu'il charge depuis une page Web. L'exemple appelle la méthode `setScrollTarget()` pour associer la barre de défilement `my_sb` au champ de texte `my_txt`.

```
/**
 Requiert :
  - Composant UIScrollBar dans la bibliothèque
*/

this.createTextField("my_txt", 10, 10, 20, 200, 100);
my_txt.wordWrap = true;

this.createClassObject(mx.controls.UIScrollBar, "my_sb", 20);

// Définition du champ de texte cible.
my_sb.setScrollTarget(my_txt);

// Dimensionnement en fonction du champ de texte.
my_sb.setSize(16, my_txt._height);

// Déplacement près du champ de texte.
my_sb.move(my_txt._x + my_txt._width, my_txt._y);

// Défilement de 2 lignes par clic d'une flèche de défilement.
my_sb.lineScrollSize = 2;
```

```
// Défilement de 5 lignes par clic d'un rail de défilement.
my_sb.pageScrollSize = 5;

// Chargement du texte à afficher et définition du gestionnaire onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_txt.text = src;
    } else {
        my_txt.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

UIScrollBar._targetInstanceName

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

scrollBarInstance._targetInstanceName

Description

Propriété : indique le nom de l'occurrence du champ de texte associé à un composant UIScrollBar. Cette propriété peut être testée et définie. Néanmoins, elle ne doit pas être utilisée pour créer une association entre un champ de texte et une barre de défilement. Remplacement par `UIScrollBar.setScrollTarget()`.

Exemple

L'exemple suivant crée une barre de défilement pour faire défiler, dans un champ de texte, du texte qu'il charge depuis une page Web. L'exemple appelle la fonction `trace()` pour afficher la valeur de la propriété `targetInstanceName`.

```
/**
 Requiert :
  - Composant UIScrollBar dans la bibliothèque
*/

this.createTextField("my_txt", 10, 10, 20, 200, 100);
my_txt.wordWrap = true;

this.createClassObject(mx.controls.UIScrollBar, "my_sb", 20);

// Définition du champ de texte cible.
my_sb.setScrollTarget(my_txt);

trace(my_sb._targetInstanceName);

// Dimensionnement en fonction du champ de texte.
my_sb.setSize(16, my_txt._height);

// Déplacement près du champ de texte.
my_sb.move(my_txt._x + my_txt._width, my_txt._y);

// Définition des propriétés de défilement.
my_sb.setScrollProperties(10, 0, 99);

// Chargement du texte à afficher et définition du gestionnaire onData.
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
    if (src != undefined) {
        my_txt.text = src;
  my_txt.condenseWhite = true;
    } else {
        my_txt.text = "Error loading text.";
    }
};
my_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

CHAPITRE 56

Classes de service Web

56

Les classes de service Web (qui se trouvent dans le package mx.services) vous permettent d'accéder aux services Web utilisant le protocole SOAP (Simple Object Access Protocol). Cette API n'est pas la même que l'API WebServiceConnector. L'API de service Web est un ensemble de classes que vous pouvez utiliser uniquement dans le code ActionScript, elle est commune à plusieurs produits Adobe. En revanche, le composant WebServiceConnector est une API appartenant uniquement à Flash : il fournit une interface ActionScript au composant WebServiceConnector visuel.

REMARQUE	Les classes de service Web sont prises en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Le tableau suivant répertorie les classes du package mx.services. Ces classes sont étroitement intégrées. Si vous découvrez ce package pour la première fois, vous pouvez en savoir plus en suivant l'ordre de ce tableau.

Classe	Description
Classe WebService	Utilisation d'un fichier WSDL (Web Service Definition Language) qui définit le service Web, élabore un nouvel objet WebService pour appeler des méthodes de service Web et gérer des rappels à partir du service Web.
Classe PendingCall	Objet renvoyé à la suite d'un appel de méthode de service Web que vous mettez en place pour gérer les résultats et les erreurs de l'appel.
Classe Log	Objet facultatif utilisé pour enregistrer l'activité associée à un objet WebService.
Classe SOAPCall	Classe améliorée qui contient des informations sur le fonctionnement du service Web et permet de contrôler certains comportements.

Disponibilité des classes de service Web à l'exécution

Afin de rendre les classes de service Web disponibles lors de l'exécution, le composant WebServiceConnector doit se trouver dans la bibliothèque de votre fichier FLA. Ce composant contient les classes d'exécution vous permettant de travailler avec les services Web.

REMARQUE	Ces classes deviennent automatiquement disponibles pour votre document Flash lorsque vous ajoutez un composant WebServiceConnector à votre fichier FLA.

Classe Log

Nom de classe ActionScript mx.services.Log

La classe Log fait partie du package mx.services et est utilisée avec la classe WebService (voir « Classe WebService », à la page 1491). Pour une présentation des classes du package mx.data.services, reportez-vous à « Classes de service Web », à la page 1467.

REMARQUE	La classe Log est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Vous pouvez créer un nouvel objet Log pour enregistrer l'activité associée à un objet WebService. Pour exécuter le code lors de l'envoi des messages à un objet Log, utilisez la fonction de rappel `Log.onLog()`. Il n'existe aucun fichier journal ; le mécanisme de consignation est celui utilisé dans la fonction de rappel `onLog()` (envoi des messages de consignation vers une instruction `trace()`, par exemple).

Le constructeur de cette classe crée un objet Log pouvant être transmis comme paramètre facultatif au constructeur WebService (voir « Classe WebService », à la page 1491).

Méthodes de la classe Log

Le tableau suivant répertorie les méthodes de la classe PendingCall.

Méthode	Description
`Log.getDateString()`	Renvoie la date et l'heure actuelles sous la forme d'une chaîne au format suivant : `mm/dd hh:mm:ss` utilisé par des messages Log.
`Log.logInfo()`	Génère un événement `Log.onLog` avec un niveau de journal désigné et un message désigné.
`Log.logDebug()`	Génère un événement `Log.onLog` avec un niveau de journal désigné `Log.DEBUG` et un message désigné.

Résumé des propriétés de l'objet Log

Le tableau suivant répertorie les propriétés de la classe PendingCall.

Propriété	Description
`Log.level`	La catégorie d'informations que vous souhaitez enregistrer dans le journal.
`Log.name`	Un nom de chaîne identifiant l'objet Log inclus dans chaque message d'événement `Log.onLog`.

Rappels de l'objet Log

Le tableau suivant présente le rappel de l'objet Log.

Rappel	Description
`Log.onLog()`	Appelé par Flash Player lorsqu'un message de consignation est envoyé à un fichier journal.

Constructeur de la classe Log

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
myWebSrvcLog = new Log([logLevel] [, logName]);
```

Paramètres

logLevel Niveau indiquant le type d'informations que vous souhaitez enregistrer dans le journal. Quatre niveaux de journal sont disponibles :

- `Log.BRIEF` Le journal enregistre les notifications d'erreur et d'événement de cycle de vie primaire. Il s'agit de la valeur par défaut.
- `Log.VERBOSE` Le journal enregistre toutes les notifications d'erreur et d'événement de cycle de vie.
- `Log.DEBUG` Le journal enregistre les événements et erreurs de mesure et de granularité fine.
- `Log.NONE` Le journal n'enregistre rien. Peut être utilisé pour désactiver temporairement des événements `Log.onLog`.

logName Nom facultatif inclus dans chaque message de consignation. Si vous utilisez plusieurs objets Log, vous pouvez utiliser le nom du journal pour identifier le journal ayant enregistré un message donné.

Valeur renvoyée

Aucune.

Description

Constructeur : crée un objet Log. Une fois l'objet Log créé, vous pouvez le transmettre à un service Web pour recevoir des messages.

Exemple

Vous pouvez appeler le constructeur `new Log` pour renvoyer un objet Log à transmettre à votre service Web :

```
// Création d'un objet Log.
import mx.services.*;
myWebSrvcLog = new Log();
myWebSrvcLog.onLog = function(msg : String) : Void
{
  myTrace(txt)
}
```

Vous transmettez ensuite cet objet Log en tant que paramètre au constructeur WebService :

```
myWebSrvc = new WebService("http://www.myco.com/info.wsdl", myWebSrvcLog);
```

A chaque exécution du code des services Web et d'envoi des messages vers l'objet Log, la fonction `onLog()` de ce dernier est appelée. Il s'agit du seul endroit possible pour placer le code qui affiche les messages de consignation si vous souhaitez les voir en temps réel.

Voici des exemples de messages de consignation :

```
7/30 15:22:43 [INFO] SOAP: Decoding PendingCall response
7/30 15:22:43 [DEBUG] SOAP: Decoding SOAP response envelope
7/30 15:22:43 [DEBUG] SOAP: Decoding SOAP response body
7/30 15:22:44 [INFO] SOAP: Decoded SOAP response into result [16 millis]
7/30 15:22:46 [INFO] SOAP: Received SOAP response from network [6469 millis]
7/30 15:22:46 [INFO] SOAP: Parsed SOAP response XML [15 millis]
7/30 15:22:46 [INFO] SOAP: Decoding PendingCall response
7/30 15:22:46 [DEBUG] SOAP: Decoding SOAP response envelope
7/30 15:22:46 [DEBUG] SOAP: Decoding SOAP response body
7/30 15:22:46 [INFO] SOAP: Decoded SOAP response into result [16 millis]
```

Log.getDateString()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`myWebSrvcLog``.getDateString()`

Paramètres

Aucun.

Valeur renvoyée

La date et l'heure actuelles sous la forme d'une chaîne au format suivant : `mm/dd hh:mm:ss`.

Description

Fonction : renvoie la date et l'heure actuelles sous la forme d'une chaîne au format suivant : `mm/dd hh:mm:ss`. Vous pouvez utiliser `Log.getDateString()` pour obtenir la date au même format fourni dans un message de consignation ou enregistrer uniquement la chaîne de date dans un gestionnaire d'événements `log.onLog` pour l'utiliser avec la gestion de journaux personnalisée.

Exemple

L'exemple suivant crée un objet Log, le transmet à un nouvel objet WebService et gère les messages de consignation à l'aide de `Log.getDateString()` pour obtenir l'heure du journal.

```
import mx.services.*;
// Création d'un objet Log.
myWebSrvcLog = new Log(Log.BRIEF, "myLog");
// Transmission de l'objet Log au service Web.
myWebService = new WebService(wsdlURI, myWebSrvcLog);
// Gestion des messages de consignation entrants.
myWebSrvcLog.onLog = function(message : String) : Void
{
  trace("A Log Event Occurred At This Time: "+ this.getDateString());
}
```

Log.logInfo()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`myWebSrvcLog``.logInfo(`*`msg`*`)`

Paramètres

`msg` Un message du type chaîne que vous souhaitez faire apparaître dans le message d'événement log résultant.

Valeur renvoyée

Aucune.

Description

Fonction : génère un message de consignation défini par le paramètre `msg` à un niveau de journal défini par le paramètre `level`. Cette méthode permet de créer vos propres événements log avec n'importe quel niveau de journalisation.

Exemple

L'exemple suivant crée un objet Log. Un événement `onLog` avec un message indiquant le début d'un nouveau journal est généré en appelant `Log.logDebug()`.

```
import mx.services.*;
// Création d'un objet Log.
myWebSrvcLog = new Log(Log.VERBOSE, "myLog");
// Handles incoming log messages.
myWebSrvcLog.onLog = function(message:String):Void  {
     trace(message);
};
myWebSrvcLog.logInfo(">>>>> New Log Started <<<<<");
// Passes the Log object to the web service.
var wsdlURI:String = "http://www.flash-mx.com/mm/tips/tips.cfc?WSDL";
var myWebService:WebService = new WebService(wsdlURI, myWebSrvcLog);
```

Log.logDebug()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`myWebSrvcLog``.logDebug(msg)`

Paramètres

`msg` Chaîne de message de consignation. La chaîne que vous indiquez dans ce paramètre apparaît comme le message de consignation dans l'événement log résultant.

Valeur renvoyée

Aucune.

Description

Fonction : génère un message de consignation contenant `msg` et l'indicateur du type de message `[debug]`. Cette méthode vous permet de créer vos propres événements log avec `[debug]` dans le message de consignation qui ne pourra être affiché qu'avec un paramètre de niveau journal `Log.DEBUG`.

La chaîne suivante est un exemple d'un message de journalisation de niveau debug généré par `Log.logDebug()`:

```
12/18 23:20:17 [DEBUG] myLog: My log message
```

Exemple

L'exemple suivant crée un objet Log. Un événement `onLog` avec un message indiquant le début d'un nouveau journal est généré en appelant `Log.logDebug()`.

```
import mx.services.*;
// Création d'un objet Log.
myWebSrvcLog = new Log(Log.DEBUG, "myLog");

// Gestion des messages de consignation entrants.
myWebSrvcLog.onLog = function(message : String) : Void
{
  trace(message);
}
// Génération d'un message de consignation avec un niveau de
// journalisation Log.DEBUG.
myWebSrvcLog.logDebug("New Log Started");
// Transmission de l'objet log au service Web.
myWebService = new WebService(wsdlURI, myWebSrvcLog);
```

Log.level

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
myLevel_Number = myWebSrvcLog.level
```

Description

Propriété : indique la catégorie d'informations que vous souhaitez enregistrer dans le journal. Quatre niveaux de journal sont disponibles :

- `Log.BRIEF` Le journal enregistre les notifications d'erreur et d'événement de cycle de vie primaire. Il s'agit de la valeur par défaut. Une propriété `Log.level` définie sur `Log.BRIEF`renvoie le chiffre 0.
- `Log.VERBOSE` Le journal enregistre toutes les notifications d'erreur et d'événement de cycle de vie. Une propriété `Log.level` définie sur `Log.VERBOSE` renvoie le chiffre 1.
- `Log.DEBUG` Le journal enregistre les événements et erreurs de mesure et de granularité fine. Une propriété `Log.level` définie sur `Log.DEBUG` renvoie le chiffre 2.
- `Log.NONE` Le journal n'enregistre rien. Peut être utilisé pour désactiver temporairement des événements `Log.onLog`. Une propriété `Log.level` définie sur `Log.NONE` renvoie le chiffre -1.

Même si vous pouvez définir cette propriété directement, la propriété `Log.level` est généralement définie en tant que paramètre lors de la création d'un objet Log. Voir « Classe Log », à la page 1468.

Exemple

L'exemple suivant crée un objet Log avec une propriété `Log.level` de `Log.DEBUG`. La propriété `Log.level` actuelle est représentée. Ensuite, la propriété `Log.level` de l'objet Log est définie sur `Log.VERBOSE`.

```
import mx.services.*;
// Création d'un objet Log.
myWebSrvcLog = new Log(Log.DEBUG, "myLog");
trace("myWebSrvcLog.level: "+ myWebSrvcLog.level);

// A présent, modification du niveau de l'objet Log.
myWebSrvcLog.level = Log.VERBOSE;
trace("myWebSrvcLog.level: "+ myWebSrvcLog.level);
```

Log.name

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myWebServiceName = *myWebSrvcLog*.name

Description

Propriété : une chaîne identifiant l'occurrence Log incluse dans tous les messages d'événement `Log.onLog`. Cette propriété peut être à la fois get et set. Elle est généralement définie lors de la création d'un objet Log. Voir « Classe Log », à la page 1468.

Exemple

L'exemple suivant crée un objet Log avec une propriété `Log.level` de `Log.VERBOSE` et un nom « `myLog` ». La propriété `Log.name` actuelle est représentée. Ensuite, la propriété `Log.name` de l'objet Log est définie sur « `myNewLogName` ».

```
import mx.services.*;
// Création d'un objet Log.
myWebSrvcLog = new Log(Log.VERBOSE, "myLog");
trace("myWebSrvcLog.level: "+ myWebSrvcLog.level);

// Définition d'un nouveau nom pour l'objet Log.
myWebSrvcLog.name = "myNewLogName";
trace("myWebSrvcLog.name: " + myWebSrvcLog.name);
```

Log.onLog()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
myWebSrvcLog.onLog = function(message)
```

Paramètres

message Message de consignation transmis au gestionnaire. Exemple :

```
"7/30 15:22:43 [INFO] SOAP: Decoding PendingCall response"
```

Valeur renvoyée

Aucune.

Description

Fonction de rappel : appelée par Flash Player lorsqu'un message de consignation est envoyé à un fichier journal. Cette fonction est l'endroit idéal pour placer le code d'enregistrement ou d'affichage des messages de consignation (commande `trace`, par exemple). (Pour plus d'informations sur la structure du journal, reportez-vous à « Classe Log », à la page 1468.)

Exemple

L'exemple suivant crée un objet Log, le transmet à un nouvel objet WebService et gère les messages de consignation.

```
import mx.services.*;
// Création d'un objet Log.
myWebSrvcLog = new Log();
// Transmission de l'objet Log au service Web.
myWebService = new WebService(wsdlURI, myWebSrvcLog);
// Gestion des messages de consignation entrants.
myWebSrvcLog.onLog = function(message : String) : Void
{
  mytrace("myWebSrvcLog.message: " + message);
}
```

Classe PendingCall

Nom de classe ActionScript mx.services.PendingCall

La classe PendingCall fait partie du package mx.services et est utilisée avec la classe WebService. Pour obtenir une vue d'ensemble des classes du package mx.services, reportez-vous à « Classes de service Web », à la page 1467.

REMARQUE	La classe PendingCall est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Vous ne créez pas d'objet PendingCall ni utilisez de fonction de constructeur mais lorsque vous appelez une méthode sur un objet WebService, la méthode WebService renvoie un objet PendingCall. Vous utilisez les fonctions de rappel `PendingCall.onResult` et `PendingCall.onFault` pour gérer la réponse asynchrone de la méthode du service Web. Si la méthode du service Web renvoie une erreur, Flash Player appelle `PendingCall.onFault` et transmet un objet SOAPFault représentant l'erreur XML SOAP renvoyée par le serveur ou le service Web. Un objet SOAPFault n'est pas directement construit par vous mais renvoyé suite à un échec. Il s'agit de la correspondance dans ActionScript du type XML SOAPFault.

Si l'invocation du service Web est réussie, Flash Player appelle `PendingCall.onResult` et transmet un objet résultat. L'objet résultat est la réponse XML du service Web décodée ou convertie dans ActionScript. Pour plus d'informations sur l'objet WebService, reportez-vous à « Classe WebService », à la page 1491.

L'objet PendingCall vous donne également accès à de nombreux paramètres de sortie lorsque la méthode de service Web renvoie plusieurs résultats. La « valeur de renvoi » à laquelle il est fait référence dans cette API est simplement le premier (ou seul) résultat ; pour accéder à tous les résultats, vous pouvez utiliser les fonctions « get output ». Par exemple, si la valeur renvoyée que vous obtenez dans le paramètre du rappel `onResult` n'est pas le seul résultat auquel vous souhaitez accéder, vous pouvez utiliser `getOutputValues()` (qui renvoie un tableau) ou `getOutputValue()` (qui renvoie une valeur individuelle) pour obtenir les valeurs décodées ActionScript.

Vous pouvez également accéder directement à l'objet SOAPParameter. L'objet SOAPParameter est un objet ActionScript doté de deux propriétés : `value` (valeur ActionScript du paramètre de sortie) et `element` (valeur XML du paramètre de sortie). Les fonctions suivantes renvoient un objet SOAPParameter ou un tableau d'objets SOAPParameter : `getOutputParameters()`, `getOutputParameterByName()` et `getOutputParameter()`.

Méthodes de la classe PendingCall

Le tableau suivant répertorie les méthodes de la classe PendingCall.

Méthode	Description
`PendingCall.getOutputParameter()`	Récupère un objet SOAPParameter par index.
`PendingCall.getOutputParameterByName()`	Récupère un objet SOAPParameter par nom.
`PendingCall.getOutputParameters()`	Récupère un tableau des objets SOAPParameter.
`PendingCall.getOutputValue()`	Récupère la valeur de sortie en fonction de l'index spécifié.
`PendingCall.getOutputValues()`	Récupère un tableau de toutes les valeurs de sortie.

Propriétés de l'objet PendingCall

Le tableau suivant répertorie les propriétés de la classe PendingCall.

Propriété	Description
`PendingCall.myCall`	Descripteur d'opération SOAPCall pour l'opération PendingCall.
`PendingCall.request`	Requête SOAP au format XML brut.
`PendingCall.response`	Réponse SOAP au format XML brut.

Rappels de l'objet PendingCall

Le tableau suivant répertorie les rappels de la classe PendingCall.

Rappel	Description
`PendingCall.onFault`	Appelé par Flash Player lorsqu'une méthode de service Web a échoué et a renvoyé une erreur.
`PendingCall.onResult`	Appelé si la méthode a réussi et a renvoyé un résultat.

PendingCall.getOutputParameter()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myPendingCall.getOutputParameter(*index*)

Paramètres

index Index du paramètre (basé sur zéro).

Valeur renvoyée

Un objet SOAPParameter ayant deux propriétés : `value` (valeur ActionScript du paramètre de sortie) et `element` (valeur XML du paramètre de sortie).

Description

Fonction : obtient un paramètre de sortie supplémentaire de l'objet SOAPParameter, contenant la valeur et l'élément XML. Les appels RPC SOAP peuvent renvoyer plusieurs paramètres de sortie. La première (ou seule) valeur renvoyée est toujours donnée dans le paramètre *result* de la fonction de rappel `onResult`, mais pour accéder aux autres valeurs renvoyées, vous devez utiliser des fonctions telles que `getOutputParameter()` et `getOutputValue()`. La fonction `getOutputParameter()` renvoie le paramètre de sortie *n*th en tant qu'objet SOAPParameter.

Exemple

En considérant le fichier descripteur SOAP ci-dessous, `getOutputParameter(1)` renverrait un objet SOAPParameter avec les paramètres `value="Hi there!"` et `element=the <outParam2> XMLNode`.

```
...
<SOAP:Body>
  <rpcResponse>
    <outParam1 xsi:type="xsd:int">54</outParam1>
    <outParam2 xsi:type="xsd:string">Hi there!</outParam2>
    <outParam3 xsi:type="xsd:boolean">true</outParam3>
  </rpcResponse>
</SOAP:Body>
...
```

Voir aussi

`PendingCall.getOutputParameterByName()`, `PendingCall.getOutputParameters()`, `PendingCall.getOutputValue()`, `PendingCall.getOutputValues()`

PendingCall.getOutputParameterByName()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myPendingCall.getOutputParameterByName(var *localName*)

Paramètres

localName Nom local du paramètre. En d'autres termes, le nom d'un élément XML, dépouillé de toute information sur l'espace de nom. Par exemple, le nom local des deux éléments suivants est `bob` :

```
<bob abc="123">
<xsd:bob def="ghi">
```

Valeur renvoyée

Un objet SOAPParameter ayant deux propriétés : `value` (valeur ActionScript du paramètre de sortie) et `element` (valeur XML du paramètre de sortie).

Description

Fonction : obtient n'importe quel paramètre de sortie en tant qu'objet SOAPParameter, contenant la valeur et l'élément XML. Les appels RPC SOAP peuvent renvoyer plusieurs paramètres de sortie. La première (ou seule) valeur renvoyée est toujours donnée dans le paramètre *result* de la fonction de rappel `onResult`, mais pour accéder aux autres valeurs renvoyées, vous devez utiliser des fonctions telles que `getOutputParameterByName()`. Cette fonction renvoie le paramètre de sortie avec le nom *localName*.

Exemple

En considérant le fichier descripteur SOAP ci-dessous, `getOutputParameterByName("outParam2")` renverrait un objet SOAPParameter avec les paramètres `value="Hi there!"` et `element=the <outParam2> XMLNode`.

```
...
<SOAP:Body>
  <rpcResponse>
    <outParam1 xsi:type="xsd:int">54</outParam1>
    <outParam2 xsi:type="xsd:string">Hi there!</outParam2>
    <outParam3 xsi:type="xsd:boolean">true</outParam3>
  </rpcResponse>
</SOAP:Body>
...
```

Voir aussi

`PendingCall.getOutputParameter()`, `PendingCall.getOutputParameters()`, `PendingCall.getOutputValue()`, `PendingCall.getOutputValues()`

PendingCall.getOutputParameters()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myPendingCall.getOutputParameters()

Paramètres

Aucun.

Valeur renvoyée

Un objet SOAPParameter ayant deux propriétés : `value` (valeur ActionScript du paramètre de sortie) et `element` (valeur XML du paramètre de sortie).

Description

Fonction : obtient des paramètres de sortie supplémentaires de l'objet SOAPParameter, contenant les valeurs et les éléments XML. Les appels RPC SOAP peuvent renvoyer plusieurs paramètres de sortie. La première (ou seule) valeur renvoyée est toujours donnée dans le paramètre *result* de la fonction de rappel `onResult`, mais pour accéder aux autres valeurs renvoyées, vous devez utiliser des fonctions telles que `getOutputParameters()` et `getOutputValues()`.

Voir aussi

`PendingCall.getOutputParameterByName()`, `PendingCall.getOutputParameter()`, `PendingCall.getOutputValue()`, `PendingCall.getOutputValues()`

PendingCall.getOutputValue()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myPendingCall.getOutputValue(var *index*)

Paramètres

index Index d'un paramètre de sortie. Le premier paramètre est l'index 0.

Valeur renvoyée

Le paramètre de sortie *n*th.

Description

Fonction : obtient la valeur ActionScript décodée d'un paramètre de sortie individuel. Les appels RPC SOAP peuvent renvoyer plusieurs paramètres de sortie. La première (ou seule) valeur renvoyée est toujours donnée dans le paramètre *result* de la fonction de rappel `onResult`, mais pour accéder aux autres valeurs renvoyées, vous devez utiliser des fonctions telles que `getOutputValue()` et `getOutputParameter()`. La fonction `getOutputValue()` renvoie le paramètre de sortie *n*th.

Exemple

En considérant le fichier descripteur SOAP ci-dessous, `getOutputValue(2)` renverrait la valeur `true`.

```
...
<SOAP:Body>
  <rpcResponse>
    <outParam1 xsi:type="xsd:int">54</outParam1>
    <outParam2 xsi:type="xsd:string">Hi there!</outParam2>
    <outParam3 xsi:type="xsd:boolean">true</outParam3>
  </rpcResponse>
</SOAP:Body>
...
```

Voir aussi

`PendingCall.getOutputParameterByName()`,`PendingCall.getOutputParameter()`, `PendingCall.getOutputParameters()`, `PendingCall.getOutputValues()`

PendingCall.getOutputValues()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

myPendingCall`.getOutputValues()`

Paramètres

Aucun.

Valeur renvoyée

Un tableau des valeurs décodées de tous les paramètres de sortie.

Description

Fonction : obtient la valeur décodée ActionScript de tous les paramètres de sortie. Les appels RPC SOAP peuvent renvoyer plusieurs paramètres de sortie. La première (ou seule) valeur renvoyée est toujours donnée dans le paramètre *result* de la fonction de rappel `onResult`, mais pour accéder aux autres valeurs renvoyées, vous devez utiliser des fonctions telles que `getOutputValues()` et `getOutputParameters()`.

Voir aussi

`PendingCall.getOutputParameterByName()`, `PendingCall.getOutputParameter()`, `PendingCall.getOutputParameters()`, `PendingCall.getOutputValue()`

PendingCall.myCall

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

PendingCall`.myCall`

Description

Propriété : l'objet SOAPCall correspondant à l'opération PendingCall. L'objet SOAPCall contient des informations sur le fonctionnement du service Web et permet de contrôler certains comportements. Pour plus d'informations, voir « Classe SOAPCall », à la page 1488.

Exemple

Le rappel `onResult` suivant suit le nom de l'opération SOAPCall.

```
callback.onResult = function(result)
{
  // Vérification du nom de mon opération.
  trace("My operation name is " + this.myCall.name);
}
```

PendingCall.onFault

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
myPendingCallObj.onFault = function(fault)
{
  // Votre code ici.
}
```

Paramètres

fault Version de l'objet ActionScript décodé de l'objet SOAPFault avec les propriétés. Si les informations relatives à l'erreur proviennent d'un serveur sous la forme de fichier XML, l'objet SOAPFault correspond à la version ActionScript décodée de cet XML.

Le type d'objet erreur renvoyé à `PendingCall.onFault` est un objet SOAPFault. Il n'est pas directement construit par vous mais renvoyé suite à un échec. Il s'agit de la correspondance dans ActionScript du type XML SOAPFault.

Propriété SOAPFault	Description
`faultcode`	Chaîne : une chaîne courte décrivant l'erreur.
`faultstring`	Chaîne : la description de l'erreur sous forme lisible par un humain.
`detail`	Chaîne : les informations spécifiques à l'application associées à l'erreur, par exemple trace de pile ou autres informations renvoyées par le moteur du service Web.
`element`	XML : l'objet XML représentant la version XML de l'erreur.
`faultactor`	Chaîne : la source de l'erreur (facultatif si aucun intermédiaire n'est impliqué).

Valeur renvoyée

Aucune.

Description

Fonction de rappel : vous fournissez cette fonction que Flash Player appelle lorsqu'une méthode du service Web a échoué et renvoyé une erreur. Le paramètre *fault* est un objet SOAPFault ActionScript.

Il s'agit d'un bon emplacement pour insérer du code qui gère tout type d'erreur, par exemple en indiquant à l'utilisateur que le serveur n'est pas disponible ou en lui conseillant de contacter le service d'assistance technique, si nécessaire.

Exemple

L'exemple suivant permet de gérer des erreurs renvoyées à partir de la méthode du service Web.

```
// Gestion de toute erreur renvoyée par l'utilisation d'une méthode de
  service Web.
myPendingCallObj = myWebService.methodName(params)
myPendingCallObj.onFault = function(fault)
{
  // Interception de l'erreur SOAP.
  DebugOutputField.text = fault.faultstring;

  // Ajout du code pour gérer toutes les erreurs, par exemple, en indiquant
  // à l'utilisateur que le serveur n'est pas disponible ou en lui
  // conseillant de contacter le service d'assistance technique.

}
```

PendingCall.onResult

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
myPendingCallObj.onResult = function(result)
{
  // Votre code ici.

}
```

Paramètres

result Version objet ActionScript décodée du résultat XML renvoyé par une méthode de service Web appelée avec`myPendingCallObj = myWebService. methodName(params)`.

Valeur renvoyée

Aucune.

Description

Fonction de rappel : vous fournissez cette fonction que Flash Player appelle lorsqu'une méthode de service Web réussit et renvoie un résultat. Le résultat est une version objet ActionScript décodée du XML renvoyé par l'opération. Dans cette fonction, incluez le code qui effectue l'action appropriée en fonction du résultat. Pour renvoyer le XML brut au lieu du résultat décodé, accédez à la propriété `PendingCall.response`.

Exemple

L'exemple suivant permet de gérer les résultats renvoyés à partir de la méthode du service Web.

```
// Gestion des résultats renvoyés par l'utilisation d'une méthode
// de service Web.
myPendingCallObj = myWebService.methodName(params)
myPendingCallObj.onResult = function(result)
{
  // Interception et gestion du résultat pour cette application.
  ResultOutputField.text = result;
}
```

Voir aussi

`PendingCall.response`

PendingCall.request

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
rawXML = myPendingCallback.request;
```

Description

Propriété : contient le formulaire XML brut de la requête actuelle envoyée avec `myPendingCallback = myWebService.methodName()`. Normalement, vous ne devez pas avoir besoin d'utiliser la propriété `PendingCall.request`, mais vous le pouvez si vous souhaitez accéder aux communications SOAP envoyées sur le réseau. Pour obtenir la version ActionScript des résultats de la requête, utilisez `myPendingCallback.onResult`.

PendingCall.response

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
rawXML = myPendingCallback.response;
```

Description

Propriété : contient le formulaire XML brut de la réponse à l'appel de la méthode de service Web le plus récent envoyé avec `myPendingCallback = myWebService.methodName()`. Normalement, vous ne devez pas avoir besoin d'utiliser la propriété `PendingCall.response`, mais vous le pouvez si vous souhaitez accéder aux communications SOAP envoyées sur le réseau. Pour obtenir la version ActionScript correspondante des résultats de la requête, utilisez `myPendingCallback.onResult`.

Classe SOAPCall

Nom de classe ActionScript mx.services.SOAPCall

La classe SOAPCall fait partie du package mx.services. Il s'agit d'une classe améliorée à utiliser avec la classe WebService (voir « Classe WebService », à la page 1491). Pour une présentation des classes du package mx.data.services, reportez-vous à « Classes de service Web », à la page 1467.

REMARQUE	La classe SOAPCall est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

L'objet SOAPCall n'est pas construit par vous. Lorsque vous appelez une méthode sur un objet WebService, l'objet WebService renvoie un objet PendingCall. Pour accéder à l'objet SOAPCall associé, utilisez `myPendingCall.myCall`.

Lorsque vous créez un objet WebService, il contient les méthodes correspondant aux opérations de l'URL du fichier WSDL que vous transmettez. Derrière les séquences, un objet SOAPCall est également créé pour chaque opération dans le fichier WSDL. L'objet SOAPCall est le descripteur de l'opération et en tant que tel, il contient toutes les informations sur cette opération (comment XML doit apparaître sur le réseau, le style d'opération, etc.). Il permet aussi de contrôler certains comportements. Vous pouvez obtenir l'objet SOAPCall pour une opération donnée à l'aide de la fonction `WebService.getCall()`. Il existe un SOAPCall unique pour chaque opération, partagé par l'ensemble des appels actifs de cette opération. Dès que vous obtenez l'objet SOAPCall, vous pouvez personnaliser le descripteur en effectuant les opérations suivantes :

- Activer/Désactiver le décodage de la réponse XML
- Activer/Désactiver le délai de conversion des tableaux SOAP en objets ActionScript
- Modifier la configuration simultanée d'une opération spécifiée

Propriétés de l'objet SOAPCall

Le tableau suivant répertorie les propriétés de l'objet SOAPCall.

Propriété	Description
`SOAPCall.concurrency`	Nombre de demandes simultanées.
`SOAPCall.doDecoding`	Active/Désactive le décodage de la réponse XML.
`SOAPCall.doLazyDecoding`	Active/Désactive le « décodage paresseux » (le délai de conversion des tableaux SOAP en objets ActionScript).

SOAPCall.concurrency

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

SOAPCall.concurrency

Description

Propriété : nombre de demandes simultanées. Les valeurs possibles sont répertoriées dans le tableau ci-dessous :

Valeur	Description
SOAPCall.MULTIPLE_CONCURRENCY	Permet plusieurs appels actifs.
SOAPCall.SINGLE_CONCURRENCY	Permet un seul appel à la fois en créant une erreur après un appel actif.
SOAPCall.LAST_CONCURRENCY	Permet un seul appel en annulant les précédents.

SOAPCall.doDecoding

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

SOAPCall.doDecoding

Description

Propriété : active le décodage de la réponse XML (`true`) ou le désactive (`false`). Par défaut, la réponse XML est convertie (décodée) en objets ActionScript. Si vous voulez uniquement le XML, définissez `SOAPCall.doDecoding` sur `false`.

SOAPCall.doLazyDecoding

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

SOAPCall.doLazyDecoding

Description

Propriété : active le « décodage paresseux » de tableaux (`true`) ou le désactive (`false`). Par défaut, un algorithme de « décodage paresseux » est utilisé pour retarder la conversion des tableaux SOAP en objets ActionScript jusqu'au dernier moment ; ceci permet aux fonctions d'être exécutées plus rapidement lors du renvoi de jeux de données volumineux. Cela signifie que tout tableau reçu à distance est un objet ArrayProxy. Lorsque vous accédez à un index particulier (`foo[5]`), cet élément est décodé automatiquement si nécessaire. Vous pouvez désactiver ce comportement (tous les tableaux sont alors entièrement décodés) en définissant `SOAPCall.doLazyDecoding` sur `false`.

Classe WebService

Nom de classe ActionScript mx.services.WebService

La classe WebService fait partie du package mx.services et est utilisée avec les classes Log, PendingCall et SOAPCall. Pour obtenir une vue d'ensemble des classes du package mx.services, reportez-vous à « Classes de service Web », à la page 1467.

REMARQUE	La classe WebService est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

La classe WebService est différente de la classe WebServiceConnector. La classe WebServiceConnector fournit une interface ActionScript au composant visuel WebServiceConnector.

L'objet WebService agit comme référence locale à un service Web distant. Lorsque vous créez un objet WebService, le fichier WSDL de définition du service Web est téléchargé, analysé et placé dans l'objet. Vous pouvez alors appeler les méthodes du service Web directement sur l'objet WebService et gérer tous les rappels en provenance du service Web. Lorsque le fichier WSDL a été traité et que l'objet WebService est prêt, le rappel `WebService.onLoad` est invoqué. S'il survient un problème au cours du chargement du fichier WSDL, le rappel `WebService.onFault` est invoqué.

Lorsque vous appelez une méthode sur un objet WebService, la valeur renvoyée est un objet de rappel. Le type d'objet du rappel renvoyé à partir de toutes les méthodes du service Web est `PendingCall`. Habituellement, ces objets ne sont pas construits par vous. Ils sont générés automatiquement, suite à la méthode *`webServiceObject.webServiceMethodName()`* appelée. Ces objets ne résultent pas de l'appel WebService qui a lieu par la suite. L'objet PendingCall représente l'appel en cours. Lorsque l'opération WebService se termine (quelques secondes après l'appel d'une méthode généralement), les différents champs de données PendingCall sont renseignés et le rappel `PendingCall.onResult` ou `PendingCall.onFault` que vous fournissez est appelé. Pour plus d'informations sur l'objet PendingCall, reportez-vous à « Classe PendingCall », à la page 1478.

Flash Player met en file d'attente tous les appels effectués avant l'analyse du fichier WSDL et tente de les exécuter après l'analyse. En effet, le fichier WSDL contient des informations nécessaires à un codage correct et à l'envoi d'une requête SOAP. Les appels de fonction que vous effectuez après analyse du fichier WSDL n'ont pas besoin d'être placés dans la file d'attente ; ils sont exécutés immédiatement. Si un appel mis en file d'attente ne correspond pas au nom de l'une des opérations définies dans le fichier WSDL, Flash Player renvoie une erreur vers l'objet de rappel que vous avez reçu lorsque vous avez lancé votre appel.

L'API WebServices, incluse dans le package mx.services, est constituée des classes WebService, Log et PendingCall.

Afin de rendre les classes de service Web disponibles lors de l'exécution, le composant WebServiceConnector doit se trouver dans la bibliothèque de votre fichier FLA. Si vous utilisez ActionScript uniquement pour accéder à un service Web lors de l'exécution, vous devez ajouter ce composant manuellement à la bibliothèque de votre document.

Méthodes de l'objet WebService

Le tableau suivant répertorie les méthodes de l'objet WebService.

Méthode	Description
`WebService.getCall()`	Obtient l'objet SOAPCall pour une opération donnée.
WebService.myMethodName()	Invoque une opération spécifique du service Web définie par WSDL.

Rappels de l'objet WebService

Le tableau suivant répertorie les rappels de l'objet WebService.

Rappel	Description
`WebService.onFault`	Appelé lorsqu'une erreur se produit au cours de l'analyse WSDL.
`WebService.onLoad`	Appelé lorsque le service Web a réussi le chargement et l'analyse de son fichier WSDL.

Types pris en charge

Les classes de services Web prennent en charge un sous-jeu de types de schéma XML (types de données) comme défini dans les tableaux ci-dessous.

Les types de données complexes et le type tableau à codage SOAP sont également pris en charge. Ils peuvent se composer d'autres types complexes, tableaux ou types de schéma XML intégrés :

- « Types numériques simples », à la page 1494
- « Types simples de date et d'heure », à la page 1494
- « Types simples de nom et de chaîne », à la page 1495
- « Type booléen », à la page 1495
- « Types d'objets », à la page 1495
- « Eléments d'objet de schéma XML pris en charge », à la page 1495

Types numériques simples

Type de schéma XML	Liaison ActionScript
`decimal`	Nombre
`integer`	Nombre
`negativeInteger`	Nombre
`nonNegativeInteger`	Nombre
`positiveInteger`	Nombre
`long`	Nombre
`int`	Nombre
`short`	Nombre
`byte`	Nombre
`unsignedLong`	Nombre
`unsignedShort`	Nombre
`unsignedInt`	Nombre
`unsignedByte`	Nombre
`float`	Nombre
`double`	Nombre

Types simples de date et d'heure

Type de schéma XML	Liaison ActionScript
`date`	Objet Date
`datetime`	Objet Date
`duration`	Objet Date
`gDay`	Objet Date
`gMonth`	Objet Date
`gMonthDay`	Objet Date
`gYear`	Objet Date
`gYearMonth`	Objet Date
`time`	Objet Date

Types simples de nom et de chaîne

Type de schéma XML	Liaison ActionScript
`string`	Chaîne ActionScript
`normalizedString`	Chaîne ActionScript
`QName`	mx.services.Qname object

Type booléen

Type de schéma XML	Liaison ActionScript
`Valeur booléenne`	Valeur booléenne

Types d'objets

Type de schéma XML	Liaison ActionScript
`Any`	Objet XML
`Complex Type`	Objet ActionScript composé de propriétés de tout type pris en charge
`Array`	Tableau ActionScript composé de tout objet ou type pris en charge

Eléments d'objet de schéma XML pris en charge

La description de schéma suivante illustre les éléments d'objet de schéma XML pris en charge :

```
schema
     complexType
          complexContent
               restriction
          sequence | simpleContent
               restriction
     element
          complexType | simpleType
```

Sécurité WebService

Les méthodes et rappels de la classe WebService sont conformes au modèle de sécurité Flash Player. Pour plus d'informations sur le modèle de sécurité de Flash Player, reportez-vous à « Fonctionnement de la sécurité » dans le guide *Formation à ActionScript 2.0 dans Adobe Flash*.

Authentification et autorisation utilisateur Les règles d'authentification et d'autorisation de l'API WebService sont les mêmes que celles employées pour toute opération réseau XML de Flash. Le protocole SOAP ne spécifie aucune méthode d'authentification et d'autorisation. Par exemple, lorsque la couche de transport HTTP sous-jacente renvoie une réponse HTTP BASIC dans les en-têtes HTTP, le navigateur répond en présentant une boîte de dialogue destinée à l'utilisateur et en ajoutant la saisie de l'utilisateur dans les en-têtes HTTP des messages suivants. Ce mécanisme existe à un niveau inférieur à celui du protocole SOAP et fait partie intégrante de l'authentification HTTP dans Flash.

Intégrité du message La sécurité des messages implique le cryptage des messages SOAP au niveau d'une couche conceptuelle située au-dessus des paquets réseau sur lesquels les messages SOAP sont livrés.

Sécurité du transport Le protocole de transport réseau sous-jacent pour les services Web SOAP Flash Player est toujours HTTP `POST`. Par conséquent, tout moyen de sécurité pouvant être appliqué à la couche de transport HTTP Flash (SSL, par exemple) est géré via des appels des services Web de Flash. SSL/HTTPS propose la forme la plus courante de sécurité du transport pour la messagerie SOAP et l'utilisation de l'authentification HTTP BASIC, associée à SSL sur la couche de transport, correspond à la méthode de sécurité la plus couramment utilisée par les sites Web.

Constructeur de la classe WebService

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
myWebServiceObject = new WebService(wsdlURI [, logObject]);
```

Paramètres

wsdlURI URI du fichier WSDL du service Web.

logObject Paramètre facultatif indiquant le nom de l'objet Log pour ce service Web. Si ce paramètre est utilisé, l'objet Log doit d'abord être créé. Pour plus d'informations, voir « Classe Log », à la page 1468.

Valeur renvoyée

Aucune.

Description

Fonction de constructeur : crée un objet WebService. Lorsque vous appelez `new WebService()`, vous fournissez l'URL d'un fichier WSDL, et Flash Player renvoie un objet WebService. Le constructeur peut accepter en option l'URI d'un proxy et un objet Log.

Si vous le souhaitez, vous pouvez utiliser deux rappels pour l'objet WebService. Flash Player appelle *webServiceObject*`.onLoad` une fois l'analyse du fichier WSDL terminée et l'objet complet. Si vous voulez exécuter du code, après l'analyse complète du fichier WSDL uniquement, vous pouvez l'insérer à cet emplacement. Vous pouvez, par exemple, placer votre premier appel de méthode de service Web dans cette fonction.

Flash Player appelle *webServiceObject*`.onFault` lorsqu'une erreur se produit lors de la recherche ou de l'analyse du fichier WSDL. Il s'agit d'un bon emplacement pour insérer du code de débogage ou du code indiquant aux utilisateurs que le serveur n'est pas disponible, qu'ils doivent retenter l'opération plus tard ou toute autre information de ce type. Pour plus d'informations, consultez les entrées individuelles correspondant à ces fonctions.

Invocation d'une opération du service Web Vous invoquez une opération du service Web en tant que méthode directement disponible sur le service Web. Par exemple, si votre service Web possède la méthode `getCompanyInfo(tickerSymbol)`, invoquez alors la méthode de la manière suivante :

```
myPendingCallObject = myWebServiceObject.getCompanyInfo(tickerSymbol);
```

Dans cet exemple, l'objet de rappel s'appelle `myPendingCallObject`. Toutes les invocations de méthode sont asynchrones et renvoient un objet de rappel de type `PendingCall`. (Le terme *asynchrone* signifie que les résultats de l'appel du service Web ne sont pas immédiatement disponibles.)

Considérez l'appel suivant :

```
x = stockService.getQuote("macr");
```

Lorsque vous effectuez cet appel, l'objet `x` n'est pas le résultat de `getQuote` ; il s'agit d'un objet PendingCall. Les résultats réels sont uniquement disponibles plus tard, à la fin de l'opération du service Web. Votre code ActionScript est averti par un appel à la fonction de rappel `onResult`.

Gestion de l'objet PendingCall Cet objet de rappel est un objet PendingCall qui vous sert à gérer les résultats et les erreurs à partir de la méthode du service Web appelée (voir « Classe PendingCall », à la page 1478). Voici un exemple :

```
MyPendingCallObject = myWebServiceObject.myMethodName(param1, ..., paramN);
MyPendingCallObject.onResult = function(result)
{
  OutputField.text = result
}
MyPendingCallObject.onFault = function(fault)
{
  DebugField.text = fault.faultCode + "," + fault.faultstring;

  // Ajout du code pour gérer toutes les erreurs, par exemple, en indiquant
  // à l'utilisateur que le serveur n'est pas disponible ou en lui
  // conseillant de contacter le service d'assistance technique.

}
```

WebService.getCall()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
getCall(var operationName)
```

Paramètres

`operationName` Opération du service Web de l'objet SOAPCall correspondant que vous souhaitez récupérer.

Valeur renvoyée

Un objet SOAPCall.

Description

Fonction : lorsque vous créez un objet WebService, il contient les méthodes correspondant aux opérations de l'URL du fichier WSDL que vous transmettez. Derrière les séquences, un objet SOAPCall est également créé pour chaque opération dans le fichier WSDL. L'objet SOAPCall est le descripteur de l'opération et en tant que tel, il contient toutes les informations sur cette opération (comment XML doit apparaître sur le réseau, le style d'opération, etc.). Il permet aussi de contrôler certains comportements. Vous pouvez obtenir l'objet SOAPCall pour une opération donnée en utilisant la méthode `getCall()`. Il existe un objet SOAPCall unique pour chaque opération, partagé par l'ensemble des appels actifs de cette opération. Dès que vous obtenez l'objet SOAPCall, vous pouvez modifier le descripteur d'opérateur à l'aide de la classe SOAPCall ; reportez-vous à « Classe SOAPCall », à la page 1488.

WebService.*myMethodName*()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
callbackObj = myWebServiceObject.myMethodName(param1, ... paramN);
```

Paramètres

param1, ... paramN Plusieurs paramètres, selon la méthode du service Web qui est appelée.

Valeur renvoyée

Un objet PendingCall auquel vous pouvez associer une fonction pour gérer les résultats et les erreurs lors de l'invocation. Pour plus d'informations, voir « Classe PendingCall », à la page 1478.

Le rappel invoqué lorsque la réponse est renvoyée de la méthode WebService est `PendingCall.onResult` ou `PendingCall.onFault`. En identifiant de manière unique vos objets de rappel, vous pouvez gérer plusieurs rappels `onResult`, comme dans l'exemple suivant :

```
myWebService = new WebService("http://www.myCompany.com/myService.wsdl");
callback1 = myWebService.getWeather("02451");
callback1.onResult = function(result)
{
  // action
}
callback2 = myWebService.getDetailedWeather("02451");
callback2.onResult = function(result)
{
  // Autre action.
}
```

Description

Méthode : invoque une opération du service Web. Vous invoquez la méthode directement sur le service Web. Par exemple, si votre service Web possède la méthode `getCompanyInfo(tickerSymbol)`, effectuez l'appel suivant :

```
myCallbackObject.myservice.getCompanyInfo(tickerSymbol);
```

Toutes les invocations sont asynchrones et renvoient un objet de rappel de type `PendingCall`.

WebService.onFault

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

```
Usage
MyWebServiceObject.onFault = function(fault)
{
  // Votre code ici.
}
```

Paramètres

`fault` Version de l'objet ActionScript décodé de l'erreur avec les propriétés. Si les informations relatives à l'erreur proviennent d'un serveur sous la forme de fichier XML, l'objet SOAPFault correspond à la version ActionScript décodée de cet XML.

Le type de l'objet erreur renvoyé aux méthodes `WebService.onFault` est un objet SOAPFault. Cet objet n'est pas directement construit par vous, mais renvoyé suite à un échec. Cet objet est une correspondance dans ActionScript du type XML SOAPFault.

Propriété SOAPFault	Description
`faultcode`	Chaîne : le QName court standard décrivant l'erreur.
`faultstring`	Chaîne ; dscription de l'erreur sous forme lisible par un humain.
`detail`	Chaîne ; informations spécifiques à l'application qui sont associées à l'erreur, par exemple trace de pile ou autres informations renvoyées par le moteur du service Web.
`element`	XML ; objet XML représentant la version XML de l'erreur.
`faultactor`	Chaîne : la source de l'erreur. Facultatif si aucun intermédiaire n'est impliqué.

Valeur renvoyée

Aucune.

Description

Fonction de rappel : appelée par Flash Player lorsque le constructeur `new WebService()` a échoué et renvoyé une erreur. Cela peut se produire lorsque le fichier WSDL ne peut pas être analysé ou lorsqu'il est introuvable. Le paramètre *`fault`* est un objet SOAPFault ActionScript.

Exemple

L'exemple suivant permet de gérer toute erreur renvoyée suite à la création de l'objet WebService.

```
MyWebServiceObject.onFault = function(fault)
{
  // Capture de l'erreur.
  DebugOutputField.text = fault.faultstring;

  // Ajout du code pour gérer toutes les erreurs, par exemple, en indiquant
  // à l'utilisateur que le serveur n'est pas disponible ou en lui
  // conseillant de contacter le service d'assistance technique.

}
```

WebService.onLoad

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
myService.onLoad = function(wsdlDocument)
{
  // Votre code ici.
}
```

Paramètres

wsdlDocument Document XML WSDL.

Valeur renvoyée

Aucune.

Description

Fonction de rappel : appelée par Flash Player lorsque l'objet WebService a réussi le chargement et l'analyse de son fichier WSDL. Si des opérations sont invoquées dans une application avant l'appel de cette fonction de rappel, elles sont mises en file d'attente en interne et ne sont pas transmises tant que le chargement du fichier WSDL n'a pas eu lieu.

Exemple

L'exemple suivant spécifie l'URI du fichier WSDL, crée un objet WebService et reçoit le document WSDL après chargement. Lorsque la bibliothèque contient les composants WebServiceClasses et Alert, ajoutez le code suivant à la première image du scénario principal :

```
import mx.controls.Alert;
import mx.services.WebService;

// Spécification de l'URI du fichier WSDL.
var wsdlURI = "http://www.flash-db.com/services/ws/companyInfo.wsdl";

// Création d'un objet de service Web.
stockService = new WebService(wsdlURI);

stockService.onLoad = function(wsdlURI){
            Alert.show("Loaded");
}
```

CHAPITRE 57

Composant WebServiceConnector

57

Le composant WebServiceConnector vous permet d'accéder aux méthodes distantes présentées par un serveur utilisant le protocole standard SOAP (Simple Object Access Protocol). Une méthode de service Web peut accepter des paramètres et renvoyer un résultat. A l'aide de l'outil de programmation Flash et du composant WebServiceConnector, vous pouvez lancer des analyses par inspection, accéder aux données d'un service Web distant et lier ces données à votre application Flash.

REMARQUE

Le composant WebServiceConnector est pris en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Une seule occurrence du composant WebServiceConnector peut être utilisée pour appeler plusieurs fois la même opération. Vous devez utiliser une occurrence différente de WebServiceConnector pour chaque opération distincte que vous souhaitez appeler.

REMARQUE

Si le composant WebServiceConnector apparaît sur la scène au cours de la programmation de l'application, il n'est pas visible dans l'application.

Pour plus d'informations sur l'utilisation des résultats de ce composant, reportez-vous à *Utilisation des schémas dans l'onglet Schéma* de *Utilisation de Flash*.

Utilisation du composant WebServiceConnector

Vous pouvez utiliser le composant WebServiceConnector pour vous connecter à un service Web et pour rendre les propriétés du service Web disponibles pour effectuer la liaison avec les propriétés des composants de l'interface utilisateur de votre application. Pour vous connecter à un service Web, vous devez tout d'abord entrer l'URL du fichier WDSL du service Web concerné. Vous pouvez entrer cette URL dans l'inspecteur des composants ou le panneau Services Web. Reportez-vous à « Connexion à des services Web à l'aide du composant WebServiceConnector » dans *Utilisation de Flash*.

Pour plus d'informations sur la connexion aux services Web, reportez-vous à « Liaison des données » dans *Utilisation de Flash*.

Paramètres de WebServiceConnector

Vous pouvez définir les paramètres de programmation suivants pour chaque occurrence du composant WebServiceConnector en utilisant l'onglet Paramètres de l'inspecteur des composants :

multipleSimultaneousAllowed Valeur booléenne qui indique si plusieurs appels peuvent avoir lieu simultanément ; la valeur par défaut est définie sur `false`. Si ce paramètre est défini sur `false`, la méthode `trigger()` n'effectue aucun appel si un appel est déjà en cours. Un événement status est provoqué, avec le code `CallAlreadyInProgress`. Si ce paramètre est défini sur `true`, l'appel a lieu.

operation Chaîne indiquant le nom d'une opération qui apparaît dans le port SOAP d'un fichier WSDL.

suppressInvalidCalls Valeur booléenne qui indique s'il faut supprimer un appel lorsque les paramètres ne sont pas valides ; la valeur par défaut est `false`. Si ce paramètre est défini sur `true`, alors la méthode `trigger()` n'effectue aucun appel si les paramètres liés aux données ne réalisent pas la validation. Un événement status est provoqué, avec le code `InvalidParams`. Si ce paramètre est défini sur `false`, l'appel a lieu en utilisant les données non valides, comme nécessaire.

WSDLURL Chaîne qui correspond à l'URL du fichier WSDL qui définit l'opération du service Web. Lorsque vous définissez l'URL au cours de la programmation, le fichier WSDL est immédiatement récupéré et analysé. Les paramètres qui en découlent et les informations de résultats sont visibles dans l'onglet Schéma de l'inspecteur des composants. La description du service est ajoutée au panneau Service Web. Par exemple, visitez le site www.flash-mx.com/mm/tips/tips.cfc?wsdl.

Flux de travaux courant du composant WebServiceConnector

La procédure suivante décrit le flux de travaux courant pour le composant WebServiceConnector.

Pour utiliser un composant WebServiceConnector :

1. Vérifiez que vos paramètres de publication sont bien définis sur ActionScript 2.0.
2. Utilisez le panneau Services Web pour entrer l'URL du fichier WSDL du service Web.
3. Ajoutez un appel à la méthode du service Web : sélectionnez la méthode, cliquez avec le bouton droit de la souris (Windows) ou en appuyant sur la touche Contrôle (Macintosh) et sélectionnez Ajouter un appel de méthode dans le menu contextuel.

   Une occurrence du composant WebServiceConnector est créée dans votre application. Le schéma du composant figure dans l'onglet Schéma de l'inspecteur des composants. Vous pouvez modifier librement ce schéma, pour lui ajouter, par exemple, d'autres paramètres de formatage ou de validation.

   REMARQUE

   Le schéma des propriétés `params` et `results` du composant est mis à jour à chaque fois que vous modifiez le paramètre `WSDLURL` ou `operation`. Cela annule les modifications que vous avez pu apporter aux autres paramètres.

4. Utilisez l'onglet Liaisons de l'inspecteur des composants pour lier les paramètres et les résultats du service Web maintenant définis dans le schéma aux composants de votre application.
5. Ajoutez un déclencheur pour lancer l'opération de liaison des données de l'une des façons suivantes :
   - Associez le comportement Déclencher une source de données à un bouton.
   - Ajoutez votre ActionScript pour appeler la méthode `trigger()` sur le composant WebServiceConnector.
   - Créez une liaison entre un paramètre de service Web et un bouton de l'interface utilisateur et définissez sa propriété sur AutoTrigger. Pour plus d'informations, reportez-vous à *Types de schéma* dans *Utilisation de Flash*.

Pour voir un exemple détaillé de connexion et d'affichage d'un service Web en utilisant le composant WebServiceConnector, reportez-vous à *Didacticiel sur le service Web : Conseils d'Adobe*.

Classe WebServiceConnector

Héritage RPC > WebServiceConnector

Nom de classe ActionScript mx.data.components.WebServiceConnector

Cette classe vous permet de vous connecter à des services Web distants en utilisant ActionScript plutôt que des occurrences de composant sur la scène. Pour utiliser la classe WebServiceConnector, vous devez ajouter une occurrence du composant WebServiceConnector à votre bibliothèque. Le composant ne doit pas être placé directement sur la scène. Vous devez importer la classe ActionScript mx.data.components.WebServiceConnector au début du script ou utiliser le nom de classe avec tous ses qualificatifs dans votre code.

Méthodes de la classe WebServiceConnector

Le tableau suivant présente la méthode de la classe WebServiceConnector.

Méthode	Description
`WebServiceConnector.trigger()`	Lance un appel vers un service Web.

Propriétés de la classe WebServiceConnector

Le tableau suivant répertorie les propriétés de la classe WebServiceConnector.

Propriété	Description
`WebServiceConnector.multipleSimultaneousAllowed`	Indique si plusieurs appels peuvent être effectués simultanément.
`WebServiceConnector.operation`	Indique le nom d'une opération qui apparaît dans le port SOAP d'un fichier WSDL.
`WebServiceConnector.params`	Spécifie les données envoyées au serveur lors de l'exécution de la prochaine opération `trigger()`.
`WebServiceConnector.results`	Identifie les données reçues du serveur suite à l'opération `trigger()`.
`WebServiceConnector.suppressInvalidCalls`	Indique si un appel doit être supprimé lorsque ses paramètres ne sont pas valides.
`WebServiceConnector.WSDLURL`	Détermine l'URL du fichier WSDL définissant l'opération du service Web.

Récapitulatif des événements de la classe WebServiceConnector

Le tableau suivant répertorie les événements de la classe WebServiceConnector.

Evénement	Description
`WebServiceConnector.result`	Diffusé lorsqu'un appel vers un service Web est réussi.
`WebServiceConnector.send`	Diffusé lorsque la méthode `trigger()` est en cours, une fois que les données de paramètres ont été collectées, mais avant qu'elles ne soient validées et que l'appel au service Web ne soit lancé.
`WebServiceConnector.status`	Diffusé lors du lancement d'un appel vers le service Web, pour informer l'utilisateur de l'état de l'opération.

WebServiceConnector.multiple SimultaneousAllowed

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

componentInstance`.multipleSimultaneousAllowed`

Description

Propriété : indique si plusieurs appels peuvent être effectués simultanément (`true`) ou non (`false`). Si cette propriété est définie sur `true`, l'appel a lieu. Si cette propriété est définie sur `false` et qu'un autre appel est déjà en cours, la méthode `WebServiceConnector.trigger()` provoque l'émission d'un événement `status` avec le code `CallAlreadyInProgress`.

Lorsque plusieurs appels ont lieu simultanément, il n'est pas garanti qu'ils se termineront dans l'ordre dans lequel ils ont été déclenchés. De plus, le navigateur et/ou le système d'exploitation peut définir des limites sur le nombre d'opérations réseau simultanées. La limite la plus probable que vous risquez de rencontrer est la suivante : le navigateur définit un nombre maximum d'URL pouvant être téléchargées simultanément. Ceci est souvent configurable dans un navigateur. Néanmoins, même dans ce cas, le navigateur doit mettre les flux continus en file d'attente sans interférer avec le comportement prévu de l'application Flash.

Exemple

L'exemple suivant permet plusieurs appels simultanés vers `myXmlUrl` :

```
myXmlUrl.multipleSimultaneousAllowed = true;
```

WebServiceConnector.operation

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
componentInstance.operation;
```

Description

Propriété : nom d'une opération qui apparaît dans le port SOAP d'un fichier WSDL.

Exemple

Cet exemple renvoie les données d'un service Web distant et calcule le délai nécessaire au service pour renvoyer les données au fichier SWF. Faites glisser un composant WebServiceConnector dans la bibliothèque, puis entrez le code suivant sur l'image 1 du scénario :

```
import mx.data.components.WebServiceConnector;

var startTime:Number;
var wscListener:Object = new Object();
wscListener.result = function(evt:Object) {
var resultTimeMS:Number = getTimer()-startTime;
trace("result loaded in "+resultTimeMS+" ms.");
trace(evt.target.results);
};
wscListener.send = function(evt:Object) {
```

```
startTime = getTimer();
};
var wsConn:WebServiceConnector = new WebServiceConnector();
wsConn.addEventListener("result", wscListener);
wsConn.addEventListener("send", wscListener);
wsConn.WSDLURL = "http://www.flash-mx.com/mm/tips/tips.cfc?wsdl";
wsConn.operation = "getTipByProduct";
wsConn.params = ["Flash"];
wsConn.suppressInvalidCalls = true;
wsConn.multipleSimultaneousAllowed = false;
wsConn.trigger();
```

WebServiceConnector.params

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

componentInstance.params

Description

Propriété : détermine les données qui seront envoyées au serveur lors de l'exécution de la prochaine opération `trigger()`. Le type de données est fixé par la description WSDL du service Web.

Lorsque vous appelez des méthodes de service Web, le type de données de la propriété `params` doit être un objet ou un tableau ActionScript :

- Si le service Web est au format document, le type de données `params` est un document XML.
- Si vous utilisez l'inspecteur Propriétés ou l'inspecteur de composants pour définir l'URL WSDL et l'opération lors de la création, vous pouvez fournir `params` sous forme d'un tableau de paramètres, dans l'ordre requis par la méthode du service Web (`[1, "hello", 2432]`, par exemple).

Exemple

L'exemple suivant définit la propriété `params` d'un composant de service Web appelé `wsc` :

```
wsc.params = [param_txt.text];
```

WebServiceConnector.result

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

componentInstance.addEventListener("result", *myListenerObject*)

Description

Evénement ; diffusé lorsqu'un appel vers un service Web est réussi.

Le paramètre du gestionnaire d'événements est un objet comportant les champs suivants :

- `type` : chaîne « `result` »
- `target` : référence à l'objet qui a provoqué l'événement (un composant WebServiceConnector, par exemple).

Vous pouvez récupérer la valeur réelle du résultat à l'aide de la propriété `results`.

Exemple

L'exemple suivant définit une fonction `res` pour l'événement `result` et affecte la fonction au gestionnaire d'événements `addEventListener` :

```
var res = function (ev) {
trace(ev.target.results);
};
wsc.addEventListener("result", res);
```

Cet exemple renvoie les données d'un service Web distant et effectue un calcul. Faites glisser un composant WebServiceConnector dans la bibliothèque, puis entrez le code suivant sur l'image 1 du scénario :

```
import mx.data.components.WebServiceConnector;
var wscListener:Object = new Object();
wscListener.result = function(evt:Object) {
  trace(evt.target.results);
};
var wsConn:WebServiceConnector = new WebServiceConnector();
wsConn.addEventListener("result", wscListener);
wsConn.WSDLURL = "http://www.flash-mx.com/mm/tips/tips.cfc?wsdl";
wsConn.operation = "getTipByProduct";
wsConn.params = ["Flash"];
wsConn.trigger();
```

WebServiceConnector.results

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
componentInstance.results
```

Description

Propriété : identifie les données reçues du serveur suite à une opération `trigger()`. Chaque composant WebServiceConnector définit la manière dont les données sont récupérées et les types valides. Ces données apparaissent lorsque l'opération RPC a réussi, comme indiqué par l'événement `result`. Elles sont disponibles jusqu'à la purge du composant ou jusqu'à l'opération RPC suivante.

Les données renvoyées peuvent être volumineuses. Vous pouvez gérer leur taille de deux manières :

- Sélectionnez un clip, un scénario ou un écran approprié en tant que parent du composant WebServiceConnector. Lorsque le parent est détruit, la mémoire de stockage de ce composant peut être utilisée pour collecter divers éléments.
- Vous pouvez affecter à tout moment la valeur `null` à cette propriété à l'aide d'ActionScript.

WebServiceConnector.send

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
componentInstance.addEventListener("send", myListenerObject)
```

Description

Evénement : diffusé pendant le traitement d'une opération `trigger()`, après la collecte des paramètres, mais avant la validation des données et le lancement de l'appel au service Web. Cet événement permet d'insérer du code destiné à modifier les paramètres avant l'appel.

Le paramètre du gestionnaire d'événements est un objet comportant les champs suivants :

- `type` : chaîne « `send` »
- `target` : référence à l'objet qui a provoqué l'événement (un composant WebServiceConnector, par exemple).

Vous pouvez récupérer ou modifier les valeurs réelles des paramètres à l'aide de la propriété `params`.

Exemple

L'exemple suivant définit une fonction `sendFunction` pour l'événement `send` et affecte la fonction au gestionnaire d'événements `addEventListener` :

```
var sendFunction = function (sendEnv) {
sendEnv.target.params = [newParam_txt.text];
};
wsc.addEventListener("send", sendFunction);
```

WebServiceConnector.status

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`componentInstance``.addEventListener("status", `*`myListenerObject`*`)`

Description

Evénement ; diffusé lors du lancement d'un appel vers le service Web, pour informer l'utilisateur de l'état de l'opération.

Le paramètre du gestionnaire d'événements est un objet comportant les champs suivants :

- `type` : chaîne « `status` »
- `target` : référence à l'objet qui a provoqué l'événement (un composant WebServiceConnector, par exemple).
- `code` : chaîne donnant le nom de la condition qui s'est produite
- `data` : objet dont le contenu dépend du code

Les codes et les données associées applicables à l'événement `status` sont les suivants :

Code	Données	Description
StatusChange	{callsInProgress:nnn}	Cet événement se produit chaque fois qu'un appel à un service Web est lancé ou se termine. L'élément *nnn* correspond au nombre d'appels en cours.
CallAlreadyInProgress	Aucune donnée	Cet événement est émis si l'opération `trigger()` est appelée, si `multipleSimultaneousAllowed` est défini sur `false` et qu'un appel est déjà en cours. Une fois que cet événement s'est produit, l'appel tenté est considéré comme terminé ; aucun événement `result` ou `send` ne se produit.
InvalidParams	Aucune donnée	Cet événement se produit si la méthode `trigger()` a déterminé que la propriété `params` ne contenait pas de données valides. Si la propriété `suppressInvalidCalls` est définie sur `true`, l'appel tenté est considéré comme terminé et aucun événement `result` ou `send` ne se produit.
WebServiceFault	{faultcode: code, faultstring: string, detail: detail}	Cet événement est provoqué si d'autres problèmes surviennent lors du traitement de l'appel. Les données constituent un objet SOAPFault. Une fois que l'événement s'est produit, l'appel tenté est considéré comme terminé ; aucun événement « `result` » ou « `send` » ne se produit. Consultez le tableau suivant pour obtenir une liste des erreurs pouvant se produire.

Voici les erreurs possibles du service Web :

Code d'erreur	Chaîne d'erreur	Détail
Timeout	Timeout while calling method xxx	
MustUnderstand	No callback for header xxx	
Server.Connection	Unable to connect to endpoint: xxx	
VersionMismatch	Request implements version: xxx Response implements version yyy	

Code d'erreur	Chaîne d'erreur	Détail
Client.Disconnected	Could not load WSDL	Impossible de charger WSDL, si actuellement en ligne, vérifier l'URI et/ou le format du WSDL xxx
Server	Faulty WSDL format	Les définitions doivent être le premier élément dans un document WSDL
Server.NoServicesInWSDL	Could not load WSDL	Aucun élément trouvé dans WSDL à xxx
WSDL.UnrecognizedNamespace	The WSDL parser had no registered document for the namespace xxxx	
WSDL.UnrecognizedBindingName	The WSDL parser couldn't find a binding named xxx in namespace yyy	
WSDL.UnrecognizedPortTypeName	The WSDL parser couldn't find a portType named xxx in namespace yyy	
WSDL.UnrecognizedMessageName	The WSDL parser couldn't find a message named xxx in namespace yyy	
WSDL.BadElement	Element xxx not resolvable	
WSDL.BadType	Type xxx not resolvable	
Client.NoSuchMethod	Couldn't find method 'xxx' in service	
yyy	yyy - errors reported from server, this depends on which server you talk to	
No.WSDLURL.Defined	The WebServiceConnector component had no WSDL URL defined	

Code d'erreur	Chaîne d'erreur	Détail
Unknown.Call.Failure	WebService invocation failed for unknown reasons	
Client.Disconnected	Could not load imported schema	Impossible de charger le schéma : si actuellement en ligne, vérifier l'URI et/ou le format du schéma à (XXXX)

Exemple

L'exemple suivant définit une fonction `fault` pour l'événement `status` et affecte la fonction au gestionnaire d'événements `addEventListener`. Dans l'exemple, l'URI du service est erroné de façon voulue afin de renvoyer une erreur de service Web (l'URL devrait être « `http://www.flash-mx.com/mm/tips/tips.cfc?wsdl` ») et un message demandant à l'utilisateur de vérifier l'URI. Lorsque la bibliothèque contient un composant WebServiceConnector, ajoutez le code suivant à la première image du scénario :

```
import mx.data.components.WebServiceConnector;

var fault = function (stat) {
  if (stat.code == "WebServiceFault"){
    trace(stat.data.faultcode);
    trace(stat.data.faultstring);
    trace(stat.data.detail);
  }
};

var wsConn:WebServiceConnector = new WebServiceConnector();
wsConn.addEventListener("status", fault);
wsConn.WSDLURL = "http://www.flasht-mx.com/mm/tips/tips.cfc?wsdl";
wsConn.operation = "getTipByProduct";
wsConn.params = ["Flash"];
wsConn.trigger();
```

WebServiceConnector.suppress InvalidCalls

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

componentInstance.suppressInvalidCalls

Description

Propriété : indique si un appel doit être supprimé lorsque ses paramètres ne sont pas valides. Si cette propriété est définie sur `true`, la méthode `trigger()` n'effectue aucun appel si les paramètres liés ne réalisent pas la validation. Un événement `status` est émis, avec le code `InvalidParams`. Si cette propriété est définie sur `false`, l'appel a lieu en utilisant les données non valides, comme nécessaire.

Exemple

Cet exemple affiche une erreur car les paramètres nécessaires ne sont pas transmis. Faites glisser un composant WebServiceConnector dans la bibliothèque, puis entrez le code suivant sur l'image 1 du scénario :

```
import mx.data.components.WebServiceConnector;
var res:Function = function (evt:Object) {
  trace(evt.target.results);
};
var stat:Function = function (error:Object) {
  switch (error.code) {
  case 'InvalidParams' :
    trace("Unable to connect to remote Web Service: "+error.code);
    break;
  case 'StatusChange' :
    break;
  default :
    trace("Error: "+error.code);
    break;
  }
};
var wsConn:WebServiceConnector = new WebServiceConnector();
wsConn.addEventListener("result", res);
wsConn.addEventListener("status", stat);
wsConn.WSDLURL = "http://www.flash-mx.com/mm/tips/tips.cfc?wsdl";
```

```
wsConn.operation = "getTipByProduct";
// wsConn.params = ["Flash"];
wsConn.suppressInvalidCalls = true;
wsConn.trigger();
```

Pour afficher un conseil au lieu d'une erreur, supprimez les commentaires de la ligne `wsConn.params = ["Flash"];`.

WebServiceConnector.trigger()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
componentInstance.trigger();
```

Description

Méthode : lance un appel vers un service Web. Chaque service Web définit exactement ce que l'appel implique. Si l'opération aboutit, ses résultats apparaissent dans la propriété `results` du service Web.

La méthode `trigger()` effectue les opérations suivantes :

1. Si des données sont liées à la propriété `params`, la méthode exécute toutes les liaisons afin de garantir que les données disponibles sont à jour. Cela provoque également la validation des données.
2. Si les données ne sont pas valides alors que le paramètre `suppressInvalidCalls` est défini sur `true`, l'opération est interrompue.
3. Si l'opération continue, l'événement `send` se produit.
4. L'appel distant réel est lancé à l'aide de la méthode de connexion indiquée (HTTP, par exemple).

Exemple

Cet exemple renvoie les données d'un service Web distant et effectue un calcul. Faites glisser un composant WebServiceConnector dans la bibliothèque, puis ajoutez le code suivant à l'image 1 du scénario :

```
import mx.data.components.WebServiceConnector;
var res:Function = function (evt:Object) {
  trace(evt.target.results);
```

```
};
var wsConn:WebServiceConnector = new WebServiceConnector();
wsConn.addEventListener("result", res);
wsConn.WSDLURL = "http://www.flash-mx.com/mm/tips/tips.cfc?wsdl";
wsConn.operation = "getTipByProduct";
wsConn.params = ["Flash"];
wsConn.suppressInvalidCalls = true;
wsConn.trigger();
```

WebServiceConnector.WSDLURL

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

componentInstance.WSDLURL

Description

Propriété : l'URL du fichier WSDL qui définit l'opération du service Web. Lorsque vous définissez cette URL au cours de la programmation, le fichier WSDL est immédiatement récupéré et analysé. Les paramètres et les résultats qui en découlent apparaissent dans l'onglet Schéma de l'inspecteur des composants. La description du service apparaît également dans le panneau Services Web.

Exemple

Cet exemple renvoie les données d'un service Web distant et effectue un calcul. Faites glisser un composant WebServiceConnector dans la bibliothèque, puis ajoutez le code suivant à l'image 1 du scénario :

```
import mx.data.components.WebServiceConnector;
var res:Function = function (evt:Object) {
  trace(evt.target.results);
};
var wsConn:WebServiceConnector = new WebServiceConnector();
wsConn.addEventListener("result", res);
wsConn.WSDLURL = "http://www.flash-mx.com/mm/tips/tips.cfc?wsdl";
wsConn.operation = "getTipByProduct";
wsConn.params = ["Flash"];
wsConn.suppressInvalidCalls = true;
wsConn.trigger();
```

CHAPITRE 58

Composant Window

58

Un composant Window affiche le contenu d'un clip dans une fenêtre dotée d'une barre de titre, d'une bordure et d'un bouton Fermer (en option).

REMARQUE	Le composant Window est pris en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Un composant Window peut être modal ou non modal. Une fenêtre modale empêche les actions de la souris ou la saisie au clavier dans des composants qui se trouvent à l'extérieur de la fenêtre. Le composant Window peut également être déplacé. L'utilisateur peut cliquer sur la barre de titre et faire glisser la fenêtre et son contenu à un autre emplacement. Le fait de faire glisser les bordures ne redimensionne pas la fenêtre.

Un aperçu en direct de chaque occurrence de Window permet de faire apparaître, pendant la programmation, les modifications qui ont été apportées à tous les paramètres, à l'exception de contentPath, dans l'inspecteur Propriétés ou l'inspecteur des composants.

Lorsque vous ajoutez le composant Window à une application, vous pouvez utiliser le panneau Accessibilité pour le rendre accessible aux lecteurs d'écran. Commencez par ajouter la ligne de code suivante pour activer l'accessibilité :

```
mx.accessibility.WindowAccImpl.enableAccessibility();
```

Quel que soit le nombre d'occurrences d'un composant, l'activation de son accessibilité ne se fait qu'une fois. Pour plus d'informations, reportez-vous à *Création de contenu accessible* dans *Utilisation de Flash*.

Utilisation du composant Window

Vous pouvez utiliser une fenêtre dans une application dès que vous avez besoin de présenter à l'utilisateur une information ou un choix prioritaire par rapport au reste de l'application. Par exemple, si vous souhaitez que l'utilisateur renseigne une fenêtre de connexion ou une fenêtre qui permet de modifier un mot de passe et d'en confirmer un nouveau.

Il existe plusieurs manières d'ajouter une fenêtre à une application. Vous pouvez faire glisser une fenêtre du panneau Composants jusqu'à la scène. Vous pouvez également appeler `createClassObject()` (voir `UIObject.createClassObject()`) pour ajouter une fenêtre dans une application. La troisième façon d'ajouter une fenêtre dans une application est d'utiliser la Classe PopUpManager. Utilisez Popup Manager pour créer des fenêtres modales qui se superposent aux autres objets de la scène. Pour plus d'informations, voir « Classe Window », à la page 1527.

Si vous utilisez Popup Manager pour insérer un composant Window dans un document, l'occurrence de Window possède son propre gestionnaire de focus (Focus Manager), qui est distinct du reste du document. Si vous n'utilisez pas PopUp Manager, l'ordre de focus dépend du contenu de la fenêtre et des autres composants du document. Pour plus d'informations sur le contrôle du focus, reportez-vous à « Classe FocusManager », à la page 745 ou à « Création de la navigation personnalisée du focus » dans *Utilisation des composants ActionScript 2.0*.

Les composants tels que Loader, ScrollPane et Window ont des événements pour déterminer à quel moment le chargement du contenu est terminé. Pour définir des propriétés sur le contenu d'un composant Loader, ScrollPane ou Window, ajoutez l'instruction de propriété dans un gestionnaire d'événements « complete », comme indiqué dans l'exemple suivant :

```
loadtest = new Object();
loadtest.complete = function(eventObject){
  content_mc._width= 100;
}
my_window.addEventListener("complete", loadtest)
```

Pour plus d'informations, voir la section « Window.complete », à la page 1535.

Paramètres de Window

Dans l'inspecteur de propriétés ou l'inspecteur de composants (Fenêtre > Inspecteur de composants), vous pouvez définir les paramètres de création suivants pour chaque occurrence Window :

closeButton indique si un bouton de fermeture est affiché (`true`) ou non (`false`). Cliquer sur le bouton de fermeture diffuse un événement `click`, mais ne ferme pas la fenêtre. Vous devez programmer un gestionnaire appelant `Window.deletePopUp()` pour fermer la fenêtre de façon explicite. Pour plus d'informations sur l'événement `click`, reportez-vous à `Window.click`.

REMARQUE	Si une fenêtre a été créée sans utiliser PopUp Manager, vous pouvez la fermer.

contentPath spécifie le contenu de la fenêtre. Il peut s'agir de l'identificateur de liaison du clip ou du nom de symbole d'un écran, d'un formulaire ou d'une diapositive qui contient le contenu de la fenêtre. Il peut également s'agir d'une URL absolue ou relative pour un fichier SWF ou JPG à charger dans la fenêtre. La valeur par défaut est `""`. Le contenu chargé est coupé pour être adapté à la fenêtre.

title indique le titre de la fenêtre.

REMARQUE	Les propriétés minHeight et minWidth sont utilisées par des routines de dimensionnement internes. Elles sont définies dans UIObject et remplacées par différents composants selon vos besoins. Ces propriétés peuvent être utilisées pour créer un gestionnaire de dispositions personnalisées. Dans le cas contraire, la définition de ces propriétés dans l'inspecteur de composants n'a aucun effet visible.

Dans l'inspecteur de composants, vous pouvez définir les paramètres supplémentaires suivants pour chaque occurrence du composant Window (Fenêtre > Inspecteur de composants) :

enabled est une valeur booléenne qui indique si le composant peut recevoir le focus et l'entrée. La valeur par défaut est `true`.

visible est une valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`). La valeur par défaut est `true`.

REMARQUE	Pour plus d'informations sur les cinq paramètres d'enveloppe suivants, reportez-vous à « Utilisation d'enveloppes avec le composant Window », à la page 1525.

skinCloseDisabled détermine le bouton de fermeture à l'état désactivé. La valeur par défaut est `CloseButtonDisabled`.

skinCloseDown détermine le bouton de fermeture à l'état enfoncé. La valeur par défaut est `CloseButtonDown`.

skinCloseOver détermine le bouton de fermeture à l'état survolé. La valeur par défaut est `CloseButtonOver`.

skinCloseUp détermine le bouton de fermeture à l'état relevé (par défaut). La valeur par défaut est `CloseButtonUp`.

skinTitleBackground détermine l'apparence de la barre de titre. La valeur par défaut est `TitleBackground`.

titleStyleDeclaration affecte le nom de la déclaration de style pour le texte du titre. La valeur par défaut est undefined : la barre de titre comporte du texte blanc, en gras. Reportez-vous à « Définition de styles personnalisés pour des groupes de composants » dans *Utilisation des composants ActionScript 2.0*.

Vous pouvez contrôler ces options et d'autres options du composant Window à l'aide des propriétés, méthodes et événements d'ActionScript. Pour plus d'informations, voir « Classe Window », à la page 1527.

Création d'une application avec le composant Window

La procédure suivante explique comment ajouter un composant Window à une application. Dans cet exemple, lorsque l'utilisateur clique sur un bouton, la fenêtre affiche une image.

Pour créer une application avec le composant Window :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Faites glisser un composant Window du panneau Composants à la bibliothèque du document actuel. Cette procédure ajoute le composant à la bibliothèque, mais pas sur la scène.
3. Faites glisser un composant Button du panneau Composants vers la scène ; dans l'inspecteur Propriétés, donnez-lui le nom d'occurrence **my_button**.

4. Ouvrez le panneau Actions et entrez le gestionnaire de clic suivant dans l'image 1 :

```
/**
 Requiert :
  - composant Button sur la scène (nom d'occurrence : my_button)
  - Composant Window dans la bibliothèque
*/
import mx.containers.Window;

var my_button:mx.controls.Button;

System.security.allowDomain("http://www.helpexamples.com");

// Création d'un objet écouteur.
var buttonListener:Object = new Object();
buttonListener.click = function(evt_obj:Object) {
 // Fenêtre d'instanciation.
 var my_win:MovieClip =
  mx.managers.PopUpManager.createPopUp(evt_obj.target, Window, true,
  {title:"Sample Image", contentPath:"http://www.helpexamples.com/
  flash/images/image1.jpg"});
 my_win.setSize(320, 240);
};
// Ajout de l'écouteur.
my_button.addEventListener("click", buttonListener);
```

Cet exemple crée une fonction `click()` que l'écouteur d'événement `buttonListener` appelle lorsque l'utilisateur clique sur le bouton **my_button**. Le gestionnaire d'événements click, `buttonListener.click()`, appelle `PopUpManager.createPopUp()` pour instancier une fenêtre qui affiche une image. Pour que la fenêtre se ferme lorsque vous cliquez sur OK ou Annuler, vous devez rédiger un autre gestionnaire.

Personnalisation du composant Window

Vous pouvez transformer un composant Window horizontalement et verticalement pendant la programmation et lors de l'exécution. Lors de la programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez `UIObject.setSize()`.

La modification de la taille de la fenêtre ne change pas la taille du bouton ou du titre. Le titre est aligné sur la gauche et la barre de fermeture sur la droite.

Utilisation de styles avec le composant Window

Un composant Window prend en charge les styles suivants :

Style	Thème	Description
`themeColor`	Halo	Couleur de base d'un composant. `"haloGreen"`, `"haloBlue"` et `"haloOrange"` sont des valeurs possibles. La valeur par défaut est `"haloGreen"`.
`backgroundColor`	Les deux	Couleur d'arrière-plan. La valeur par défaut est blanc pour le thème Halo et 0xEFEBEF (gris clair) pour le thème Sample.
`borderStyle`	Les deux	Le composant Window utilise une occurrence de RectBorder en tant que bordure et répond aux styles définis sur cette classe. Voir « Classe RectBorder », à la page 1111. Le composant Window présente un style de bordure propre au composant, « `default` » avec le thème Halo et « `outset` » avec le thème Sample.
`color`	Les deux	Couleur du texte. La valeur par défaut est 0x0B333C pour le thème Halo et vide pour le thème Sample.
`disabledColor`	Les deux	Couleur du texte lorsque le composant est désactivé. La couleur par défaut est 0x848384 (gris foncé).
`embedFonts`	Les deux	Valeur booléenne qui indique si la police spécifiée dans `fontFamily` est de type intégré. Ce style doit être défini sur `true` si `fontFamily` fait référence à une police intégrée. Dans le cas contraire, la police intégrée n'est pas utilisée. Si ce style est défini sur `true` et que `fontFamily` ne fait pas référence à une police intégrée, aucun texte n'est affiché. La valeur par défaut est `false`.
`fontFamily`	Les deux	Nom de la police du texte. La valeur par défaut est `"_sans"`.
`fontSize`	Les deux	Taille de la police, en points. La valeur par défaut est 10.
`fontStyle`	Les deux	Style de police : `"normal"` ou `"italic"`. La valeur par défaut est `"normal"`.
`fontWeight`	Les deux	Epaisseur de la police : `"none"` ou `"bold"`. La valeur par défaut est `"none"`. Tous les composants peuvent également accepter la valeur `"normal"` au lieu de `"none"` pendant un appel à la méthode `setStyle()`, mais les appels suivants à `getStyle()` renvoient `"none"`.

Style	Thème	Description
`textAlign`	Les deux	Alignement du texte : « `left` », « `right` » ou « `center` ». La valeur par défaut est `"left"`.
`textDecoration`	Les deux	Décoration du texte : `"none"` ou `"underline"`. La valeur par défaut est `"none"`.
`textIndent`	Les deux	Nombre indiquant le retrait du texte. La valeur par défaut est 0.

Vous pouvez définir les styles de texte sur le composant Window proprement dit, ou dans la déclaration de style de classe `_global.styles.windowStyles` (styles de texte uniquement, pas les autres styles tels que themeColor ou backgroundColor qui proviennent de la déclaration de style de classe `_global.styles.Window`). Ceci présente l'avantage suivant : les paramètres de style ne se propagent pas vers les composants enfant lors de l'héritage de style.

L'exemple suivant démontre comment mettre le titre d'un composant Window en italique sans que ce paramètre ne se propage vers les composants enfant.

```
import mx.containers.Window;
_global.styles.windowStyles.setStyle("fontStyle", "italic");
createClassObject(Window, "window", 1, {title: "A Window"});
```

Cet exemple définit la propriété avant d'instancier la fenêtre au moyen de `createClassObject()`. Pour que les styles prennent effet, ils doivent être définis avant la création de la fenêtre.

Utilisation d'enveloppes avec le composant Window

Le composant Window utilise des enveloppes pour l'arrière-plan de son titre et son bouton Fermer et une occurrence de RectBorder pour la bordure. Les enveloppes de Window se trouvent dans le dossier Flash UI Components 2/Themes/ MMDefault/Window Assets dans chacun des fichiers de thème. Pour plus d'informations, reportez-vous à « A propos de l'application d'enveloppes aux composants » dans *Utilisation des composants ActionScript 2.0*. Pour plus d'informations sur la classe RectBorder et son utilisation pour personnaliser des bordures, reportez-vous à « Classe RectBorder », à la page 1111.

L'enveloppe de l'arrière-plan du titre est toujours affichée. La hauteur de l'arrière-plan est déterminée par les graphiques de l'enveloppe. La largeur de l'enveloppe est définie par le composant Window en fonction de la taille de l'occurrence de Window. Les enveloppes de fermeture sont affichées lorsque la propriété `closeButton` est définie sur `true` et lorsqu'un état change résulte de l'interaction de l'utilisateur.

Un composant Window utilise les propriétés d'enveloppe suivantes :

Propriété	Description
`skinTitleBackground`	La barre de titre. La valeur par défaut est `TitleBackground`.
`skinCloseUp`	Le bouton Fermer. La valeur par défaut est `CloseButtonUp`.
`skinCloseDown`	Le bouton Fermer à l'état enfoncé. La valeur par défaut est `CloseButtonDown`.
`skinCloseDisabled`	Le bouton Fermer à l'état désactivé. La valeur par défaut est `CloseButtonDisabled`.
`skinCloseOver`	Le bouton Fermer à l'état survolé. La valeur par défaut est `CloseButtonOver`.

L'exemple suivant démontre comment créer un symbole de clip à utiliser comme arrière-plan de titre.

Pour définir le titre d'un composant Window sur un symbole de clip personnalisé :

1. Sélectionnez Fichier > Nouveau et choisissez Fichier Flash (ActionScript 2.0).
2. Créez un symbole en choisissant Insertion > Nouveau symbole.
3. Nommez-le `TitleBackground`.
4. Si l'écran avancé n'est pas affiché, cliquez sur le bouton Avancé.
5. Sélectionnez Exporter pour ActionScript.

   L'identifiant est automatiquement renseigné avec `TitleBackground`.
6. Définissez la valeur de la classe sur `mx.skins.SkinElement`.

   SkinElement est une classe simple qui peut être utilisée pour tous les éléments d'enveloppe qui n'assurent pas leur propre implémentation dans ActionScript. Elle apporte les fonctionnalités de mouvement et de dimensionnement requises par l'infrastructure des composants.
7. Assurez-vous que l'option Exporter dans la première image est bien activée, puis cliquez sur OK.
8. Ouvrez le nouveau symbole pour l'édition.
9. Servez-vous des outils de dessin pour créer une case rouge entourée d'une ligne noire.
10. Définissez le style de bordure sur Filet.

11. Définissez la case, et sa bordure, de sorte qu'elle soit positionnée au point (0,0), avec une largeur de 100 et une hauteur de 22.

    Le composant Window définit alors la largeur appropriée pour l'enveloppe, mais utilise la hauteur existante pour le titre.

12. Cliquez sur le bouton Précédent pour revenir au scénario principal.

13. Faites glisser un composant Window sur la scène.

14. Sélectionnez Contrôle > Tester l'animation.

Classe Window

Héritage MovieClip > Classe UIObject > Classe UIComponent > View > ScrollView > Window

Nom de classe ActionScript mx.containers.Window

Les propriétés de la classe Window permettent d'effectuer les opérations suivantes au moment de l'exécution : définir le titre, ajouter un bouton Fermer et définir le contenu d'affichage. La définition d'une propriété de la classe Window avec ActionScript annule le paramètre du même nom défini dans le panneau de l'inspecteur Propriétés ou des composants.

La meilleure façon d'instancier une fenêtre est d'appeler `PopUpManager.createPopUp()`. Cette méthode crée une fenêtre qui peut être modale (chevauchant et désactivant des objets existants dans une application) ou non modale. Le code suivant, par exemple, crée une occurrence de fenêtre modale (ce que dénote le dernier paramètre) :

```
var newWindow = PopUpManager.createPopUp(this, Window, true);
```

Flash simule la modalité en créant une grande fenêtre transparente sous le composant Window. En raison de la façon dont les fenêtres transparentes apparaissent, les objets qui se trouvent en dessous peuvent apparaître légèrement estompés. Vous pouvez définir l'effet de transparence en modifiant la valeur `_global.style.modalTransparency` entre 0 (complètement transparent) et 100 (opaque). Si vous faites en sorte que la fenêtre soit partiellement transparente, vous pouvez également définir la couleur de la fenêtre en modifiant l'enveloppe Modal dans le thème par défaut.

Si vous utilisez `PopUpManager.createPopUp()` pour créer une fenêtre modale, vous devez appeler `Window.deletePopUp()` pour la supprimer afin que la fenêtre transparente soit également supprimée. Par exemple, si vous utilisez le bouton Fermer dans la fenêtre, vous écrirez le code suivant :

```
var win = PopUpManager.createPopUp(_root, Window, true,
  {closeButton:true});
function click(evt){
  evt.target.deletePopUp();
}
win.addEventListener("click", this);
```

Le code n'interrompt pas l'exécution lorsqu'une fenêtre modale est créée. Dans d'autres environnements (Microsoft Windows par exemple), si vous créez une fenêtre modale, les lignes de code qui suivent la création de la fenêtre ne sont pas exécutées tant que la fenêtre n'est pas fermée. Dans Flash, les lignes de code sont exécutées après la création de la fenêtre et avant sa fermeture.

Toutes les classes de composants ont une propriété `version` qui correspond à une propriété de classe. Les propriétés de classe ne sont disponibles que pour la classe elle-même. La propriété `version` renvoie une chaîne qui indique la version du composant. Pour accéder à cette propriété, utilisez le code suivant :

```
trace(mx.containers.Window.version);
```

REMARQUE

Le code `trace(myWindowInstance.version);` renvoie `undefined`.

Récapitulatif des méthodes de la classe Window

Le tableau suivant présente la méthode de la classe Window.

Méthode	Description
`Window.deletePopUp()`	Supprime une occurrence de fenêtre créée par `PopUpManager.createPopUp()`.

Méthodes héritées de la classe UIObject

Le tableau suivant répertorie les méthodes de la classe Window héritées de la classe UIObject. Pour appeler ces méthodes à partir de l'objet Window, utilisez le formulaire *`WindowInstance.methodName`*.

Méthode	Description
`UIObject.createClassObject()`	Crée un objet dans la classe spécifiée.
`UIObject.createObject()`	Crée un sous-objet dans un objet.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.doLater()`	Appelle une fonction lorsque les paramètres ont été définis dans les inspecteurs des propriétés et des composants.
`UIObject.getStyle()`	Obtient la propriété de style de l'objet ou de la déclaration de style.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné dans le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image active.
`UIObject.setSize()`	Redimensionne l'objet à la taille demandée.
`UIObject.setSkin()`	Définit une enveloppe dans l'objet.
`UIObject.setStyle()`	Définit la propriété de style sur l'objet ou la déclaration de style.

Méthodes héritées de la classe UIComponent

Le tableau suivant répertorie les méthodes de la classe Window héritées de la classe UIComponent. Pour appeler ces méthodes à partir de l'objet Window, utilisez le formulaire *`WindowInstance.methodName`*.

Méthode	Description
`UIComponent.getFocus()`	Renvoie une référence à l'objet ayant le focus.
`UIComponent.setFocus()`	Attribue le focus à l'occurrence de composant.

Récapitulatif des propriétés de la classe Window

Le tableau suivant répertorie les propriétés de la classe Window.

Propriété	Description
`Window.closeButton`	Indique si la barre de titre comprend un bouton Fermer (`true`) ou non (`false`).
`Window.content`	Référence au contenu (clip racine) de la fenêtre (lecture seule).
`Window.contentPath`	Définit le nom du contenu à afficher dans la fenêtre.
`Window.title`	Texte qui apparaît dans la barre de titre.
`Window.titleStyleDeclaration`	Déclaration de style qui formate le texte dans la barre de titre.

Propriétés héritées de la classe UIObject

Le tableau suivant répertorie les propriétés de la classe Window héritées de la classe UIObject. Pour accéder à ces propriétés à partir de l'objet Window, utilisez le formulaire *`WindowInstance.propertyName`*.

Propriété	Description
`UIObject.bottom`	Lecture seule ; position du bord inférieur de l'objet par rapport au bord inférieur de son parent.
`UIObject.height`	Lecture seule ; hauteur de l'objet, en pixels.
`UIObject.left`	Lecture seule ; bord gauche de l'objet, en pixels.
`UIObject.right`	Lecture seule. Position du bord droit de l'objet par rapport au bord droit de son parent.
`UIObject.scaleX`	Nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.
`UIObject.scaleY`	Nombre indiquant le facteur de redimensionnement dans la direction *y* de l'objet par rapport à son parent.
`UIObject.top`	Lecture seule ; renvoie la position du bord supérieur de l'objet par rapport à son parent.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	Lecture seule ; largeur de l'objet, en pixels.
`UIObject.x`	Lecture seule ; bord gauche de l'objet, en pixels.
`UIObject.y`	Lecture seule ; bord supérieur de l'objet, en pixels.

Propriétés héritées de la classe UIComponent

Le tableau suivant répertorie les propriétés de la classe Window héritées de la classe UIComponent. Pour accéder à ces propriétés à partir de l'objet Window, utilisez le formulaire *`WindowInstance.propertyName`*.

Propriété	Description
`UIComponent.enabled`	Indique si le composant peut recevoir le focus et la saisie.
`UIComponent.tabIndex`	Nombre indiquant l'ordre de tabulation pour un composant dans un document.

Evénements de la classe Window

Le tableau suivant répertorie les événements de la classe Window.

Evénement	Description
`Window.click`	Diffusé lorsque le bouton de fermeture est relâché.
`Window.complete`	Diffusé à la création d'une fenêtre.
`Window.mouseDownOutside`	Diffusé lorsque l'utilisateur relâche le bouton de sa souris hors de la fenêtre modale.

Evénements hérités de la classe UIObject

Le tableau suivant répertorie les événements de la classe Window hérités de la classe UIObject.

Evénement	Description
`UIObject.draw`	Diffusé lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.hide`	Diffusé lorsqu'un objet passe de l'état visible à l'état invisible.
`UIObject.load`	Diffusé lorsque des sous-objets sont créés.
`UIObject.move`	Diffusé lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusé lorsqu'un objet a été redimensionné.
`UIObject.reveal`	Diffusé lorsqu'un objet passe de l'état invisible à l'état visible.
`UIObject.unload`	Diffusé lorsque les sous-objets sont purgés.

Evénements hérités de la classe UIComponent

Le tableau suivant répertorie les événements de la classe Window hérités de la classe UIComponent.

Evénement	Description
`UIComponent.focusIn`	Diffusé lorsqu'un objet reçoit le focus.
`UIComponent.focusOut`	Diffusé lorsqu'un objet perd le focus.
`UIComponent.keyDown`	Diffusé lorsqu'une touche est enfoncée.
`UIComponent.keyUp`	Diffusé lorsqu'une touche est relâchée.

Window.click

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.click = function(eventObject:Object) {
  // ...
};
windowInstance.addEventListener("click", listenerObject);
```

Utilisation 2 :

```
on (click) {
  // ...
}
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque l'utilisateur clique sur le bouton Fermer (puis relâche le bouton de la souris).

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence de composant (*windowInstance*) distribue un événement (dans ce cas, `click`) qui est géré par une fonction (appelée aussi *gestionnaire*) sur un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée au déclenchement de l'événement. Lors de son déclenchement, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. L'objet événement est doté de propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence Window. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence du composant. Par exemple, le code suivant, associé à l'occurrence Window `myWindow`, envoie « _level0.myWindow » vers le panneau Sortie :

```
on(click){
  trace(this);
}
```

Exemple

L'exemple suivant crée une fenêtre modale avec un bouton Fermer. Il définit un gestionnaire d'événements click qui appelle la méthode `click()` pour supprimer la fenêtre lorsque l'utilisateur clique sur le bouton. Faites glisser un composant Window du panneau Composants à la bibliothèque du document en cours, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Window dans la bibliothèque
*/

import mx.managers.PopUpManager;
import mx.containers.Window;

System.security.allowDomain("http://www.flash-mx.com");
```

```
var my_win:MovieClip = PopUpManager.createPopUp(this, Window, true,
  {closeButton:true, contentPath:"http://www.flash-mx.com/images/
  image1.jpg"});
var winListener:Object = new Object();
winListener.click = function() {
 my_win.deletePopUp();
};
my_win.addEventListener("click", winListener);
```

Voir aussi

`EventDispatcher.addEventListener()`, `Window.closeButton`

Window.closeButton

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

`windowInstance``.closeButton`

Description

Propriété : valeur booléenne qui indique si la barre de titre doit avoir un bouton de fermeture (`true`) ou non (`false`). Cette propriété doit être définie dans le paramètre *`initObject`* de la méthode `PopUpManager.createPopUp()`. La valeur par défaut est `false`.

Cliquer sur le bouton de fermeture diffuse un événement `click`, mais ne ferme pas la fenêtre. Vous devez programmer un gestionnaire appelant `Window.deletePopUp()` pour fermer la fenêtre de façon explicite. Pour plus d'informations sur l'événement `click`, reportez-vous à `Window.click`.

Exemple

L'exemple suivant crée une fenêtre contextuelle et définit la propriété `closeButton` pour y ajouter un bouton de fermeture. Faites glisser un composant Window du panneau Composants jusqu'à la bibliothèque du document actif, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Window dans la bibliothèque
*/

import mx.managers.PopUpManager;
import mx.containers.Window;

System.security.allowDomain("http://www.flash-mx.com");

var my_win:MovieClip = PopUpManager.createPopUp(this, Window, true,
  {closeButton:true, contentPath:"http://www.flash-mx.com/images/
  image1.jpg"});
```

Voir aussi

`PopUpManager.createPopUp()`, `Window.click`

Window.complete

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

```
listenerObject = new Object();
listenerObject.complete = function(eventObject){
  ...
}
windowInstance.addEventListener("complete", listenerObject)
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsqu'une fenêtre est créée. Utilisez cet événement pour dimensionner une fenêtre afin de contenir son contenu.

Une occurrence de composant (*windowInstance*) distribue un événement (ici, `complete`) qui est géré par une fonction (également appelée *gestionnaire*) sur un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée au déclenchement de l'événement. Lors de son déclenchement, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. L'objet événement est doté de propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Exemple

L'exemple suivant crée une fenêtre, puis définit un gestionnaire `complete` qui redimensionne la fenêtre en fonction de son contenu. Faites glisser un composant Window du panneau Composants jusqu'à la bibliothèque du document actif, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Window dans la bibliothèque
*/

import mx.managers.PopUpManager;
import mx.containers.Window;

System.security.allowDomain("http://www.flash-mx.com");

var my_win:MovieClip = PopUpManager.createPopUp(this, Window, true,
  {closeButton:true, contentPath:"http://www.flash-mx.com/images/
  image1.jpg"});
var winListener:Object = new Object();
winListener.click = function(evt_obj:Object) {
 my_win.deletePopUp();
};
winListener.complete = function(evt_obj:Object) {
 my_win.setSize(my_win.content._width, my_win.content._height + 25);
}
my_win.addEventListener("click", winListener);
my_win.addEventListener("complete", winListener);
```

Voir aussi

`EventDispatcher.addEventListener()`

Window.content

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

windowInstance.content

Description

Propriété en lecture seule : référence au contenu (clip racine) de la fenêtre. Cette propriété renvoie un objet MovieClip. Lorsque vous associez un symbole de la bibliothèque, la valeur par défaut est une occurrence de ce symbole. Lorsque vous chargez du contenu à partir d'une URL, la valeur par défaut est `undefined` jusqu'à ce que le chargement commence.

Exemple

L'exemple suivant crée une fenêtre, puis définit un gestionnaire `complete` qui redimensionne la fenêtre en fonction de son contenu. Il utilise la propriété `content` pour faire référence à la largeur du contenu du clip de la fenêtre. Faites glisser un composant Window du panneau Composants jusqu'à la bibliothèque du document actif, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Window dans la bibliothèque
*/

import mx.managers.PopUpManager;
import mx.containers.Window;

System.security.allowDomain("http://www.flash-mx.com");

var my_win:MovieClip = PopUpManager.createPopUp(this, Window, true,
  {closeButton:true, contentPath:"http://www.flash-mx.com/images/
  image1.jpg"});
var winListener:Object = new Object();
winListener.click = function(evt_obj:Object) {
 my_win.deletePopUp();
};
winListener.complete = function(evt_obj:Object) {
 my_win.setSize(my_win.content._width, my_win.content._height + 25);
}
my_win.addEventListener("click", winListener);
my_win.addEventListener("complete", winListener);
```

Window.contentPath

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

windowInstance.contentPath

Description

Propriété : définit le nom du contenu à afficher dans la fenêtre. Cette valeur peut être l'identificateur de liaison d'un clip de la bibliothèque ou une URL absolue ou relative du fichier SWF ou JPEG à charger. La valeur par défaut est "" (une chaîne vide).

Exemple

L'exemple suivant crée une fenêtre et utilise la propriété `contentPath` pour spécifier l'emplacement de l'image à afficher dans la fenêtre. Faites glisser un composant Window du panneau Composants jusqu'à la bibliothèque du document actif, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Window dans la bibliothèque
*/

import mx.managers.PopUpManager;
import mx.containers.Window;

System.security.allowDomain("http://www.flash-mx.com");

// Création d'une fenêtre.
var my_win:MovieClip = PopUpManager.createPopUp(this, Window, true, {
  contentPath:"http://www.flash-mx.com/images/image2.jpg"});
```

Window.deletePopUp()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

windowInstance.deletePopUp()

Paramètres

Aucun.

Valeur renvoyée

Aucune.

Description

Méthode : supprime l'occurrence de Window et supprime l'état modal. Cette méthode peut uniquement être appelée sur des occurrences de Window qui ont été créées par PopUpManager.createPopUp().

Exemple

L'exemple suivant crée une fenêtre modale, puis définit un gestionnaire d'événements clic qui appelle la fonction deletePopUp() pour la supprimer. Faites glisser un composant Window du panneau Composants jusqu'à la bibliothèque du document actif, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Window dans la bibliothèque
*/

import mx.managers.PopUpManager;
import mx.containers.Window;

System.security.allowDomain("http://www.flash-mx.com");

var my_win:MovieClip = PopUpManager.createPopUp(this, Window, true,
  {closeButton:true, contentPath:"http://www.flash-mx.com/images/
  image1.jpg"});
var winListener:Object = new Object();
winListener.click = function() {
 my_win.deletePopUp();
};
my_win.addEventListener("click", winListener);
```

Window.mouseDownOutside

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

Utilisation 1 :

```
var listenerObject:Object = new Object();
listenerObject.mouseDownOutside = function(eventObject:Object) {
  // ...
};
windowInstance.addEventListener("mouseDownOutside", listenerObject);
```

Utilisation 2 :

```
on (mouseDownOutside) {
  // ...
}
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque l'utilisateur clique sur le bouton de la souris (puis le relâche) en dehors de la fenêtre modale. Cet événement est rarement utilisé, mais vous pouvez vous en servir pour fermer une fenêtre si l'utilisateur essaie d'interagir avec un élément situé à l'extérieur de cette fenêtre.

Le premier exemple d'utilisation fait appel à un modèle dispatcher(diffuseur)/écouteur d'événement. Une occurrence de composant (*windowInstance*) distribue un événement (dans ce cas, `mouseDownOutside`) qui est géré par une fonction (appelée aussi *gestionnaire*) sur un objet écouteur (*listenerObject*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet écouteur. La méthode est appelée au déclenchement de l'événement. Lors de son déclenchement, il transmet automatiquement un objet événement (*eventObject*) à la méthode d'objet écouteur. L'objet événement est doté de propriétés qui contiennent des informations sur l'événement. Vous pouvez utiliser ces propriétés pour écrire le code qui traitera l'événement. Pour finir, vous appelez la méthode `EventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur est appelé.

Pour plus d'informations, voir « Classe EventDispatcher », à la page 515.

Le second exemple d'utilisation fait appel à un gestionnaire `on()` et doit être associé directement à une occurrence Window. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence du composant. Par exemple, le code suivant, associé à l'occurrence Window `myWindowComponent`, envoie « _level0.myWindowComponent » vers le panneau Sortie :

```
on (mouseDownOutside) {
  trace(this);
}
```

Exemple

L'exemple suivant crée une occurrence Window et définit un gestionnaire `mouseDownOutside` qui affiche un message si l'utilisateur clique en dehors de la fenêtre. Faites glisser un composant Window du panneau Composants jusqu'à la bibliothèque du document actif, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Window dans la bibliothèque
*/

import mx.managers.PopUpManager;
import mx.containers.Window;

System.security.allowDomain("http://www.flash-mx.com");

// Création d'une fenêtre.
var my_win:MovieClip = PopUpManager.createPopUp(this, Window, true,
  undefined, true);

// Création d'un objet écouteur.
var winListener:Object = new Object();
winListener.mouseDownOutside = function(evt_obj:Object)
{
    trace("mouseDownOutside event triggered.");
}
// Ajout de l'écouteur.
my_win.addEventListener("mouseDownOutside", winListener);
```

Voir aussi

`EventDispatcher.addEventListener()`

Window.title

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

windowInstance.title

Description

Propriété : chaîne correspondant au texte de la barre de titre. La valeur par défaut est "" (une chaîne vide).

Exemple

L'exemple suivant crée une fenêtre contextuelle et utilise la propriété `title` pour définir le titre sur « Hello World ». Faites glisser un composant Window du panneau Composants jusqu'à la bibliothèque du document actif, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Window dans la bibliothèque
*/

import mx.managers.PopUpManager
import mx.containers.Window

// Création d'une fenêtre.
var my_win:MovieClip = PopUpManager.createPopUp(this, Window, true);

// Définition des attributs de la fenêtre.
my_win.title = "Hello World!";
my_win.setSize(200, 100);
my_win.move(20, 20);
```

Window.titleStyleDeclaration

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX 2004.

Utilisation

windowInstance.titleStyleDeclaration

Description

Propriété : chaîne indiquant la déclaration de style qui formate la barre de titre d'une fenêtre. La valeur par défaut est `undefined`, ce qui correspond à gras et texte blanc.

Exemple

L'exemple suivant crée une déclaration de style CSS pour transformer le texte en format 14 points, en italique et souligné. Il utilise la propriété `titleStyleDeclaration` pour appliquer ce style au titre de la fenêtre contextuelle qu'il crée. Faites glisser un composant Window du panneau Composants à la bibliothèque du document en cours, puis ajoutez le code suivant à l'image 1 :

```
/**
 Requiert :
  - Composant Window dans la bibliothèque
*/

import mx.styles.CSSStyleDeclaration
import mx.managers.PopUpManager
import mx.containers.Window

// Création d'un CSSStyleDeclaration nommé TitleStyles.
// et énumération avec la liste des styles globale
_global.styles.TitleStyles = new CSSStyleDeclaration();

// Personnalisation des styles.
_global.styles.TitleStyles.fontStyle = "italic";
_global.styles.TitleStyles.textDecoration = "underline";
_global.styles.TitleStyles.color = 0xff0000;
_global.styles.TitleStyles.fontSize = 14;

// Création d'une fenêtre.
var my_win:MovieClip = PopUpManager.createPopUp(this, Window, true,
  {closeButton:true, titleStyleDeclaration:"TitleStyles"});
```

```
// Définition des attributs de la fenêtre.
my_win.title = "Testing Styles";
my_win.setSize(200, 100);
my_win.move(20, 20);

// Création d'un objet écouteur.
var winListener:Object = new Object();
winListener.click = function(evt_obj:Object) {
 trace("closing window");
 evt_obj.target.deletePopUp();
};
// Ajout de l'écouteur.
my_win.addEventListener("click", winListener);
```

Pour plus d'informations, reportez-vous à « Utilisation de styles pour personnaliser la couleur et le texte des composants » dans *Utilisation des composants ActionScript 2.0*.

CHAPITRE 59

Composant XMLConnector 59

Le composant XMLConnector vous permet de lire ou d'écrire des documents XML à l'aide d'opérations HTTP `GET` et `POST`. Il agit en tant que connecteur entre les autres composants et des sources de données XML externes.

REMARQUE	Le composant XMLConnector est pris en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Le composant XMLConnector communique avec d'autres composants de votre application en utilisant soit des fonctions de liaison de données de l'environnement de programmation Flash Professional, soit du code ActionScript. Le composant XMLConnector possède des propriétés, des méthodes et des événements, mais n'est pas visible lors de l'exécution.

Utilisation du composant XMLConnector

Le composant XMLConnector permet à l'application d'accéder à toute source de données externe renvoyant ou recevant des données XML via HTTP. Le moyen le plus simple de se connecter à une source de données XML externe et d'utiliser les paramètres et les résultats de cette source de données consiste à spécifier un *schéma*, c'est-à-dire la structure du document XML qui identifie les éléments de données du document sur lesquels vous pouvez créer des liaisons.

Pour plus d'informations sur l'utilisation du composant XMLConnector, reportez-vous à *Connexion aux données XML à l'aide du composant XMLConnector* dans *Utilisation de Flash*.

Paramètres de XMLConnector

Vous pouvez définir les paramètres de programmation suivants pour chaque occurrence de XMLConnector dans l'onglet Paramètres de l'inspecteur des composants :

URLChaîne pointant vers une source de données XML externe.

direction Chaîne qui définit l'opération HTTP à effectuer lorsque la méthode `XMLConnector.trigger()` est appelée. Ce paramètre peut prendre les valeurs suivantes : « `send` », « `receive` » ou « `send/receive` ».

Une valeur « `send` » signifie que les données XML sont envoyées (via HTTP POST) à l'URL, mais Flash ignore les données qui sont renvoyées. La propriété `XMLConnector.results` n'est jamais définie et aucun événement `result` ne se produit.

Une valeur « `receive` » signifie qu'aucune donnée n'est envoyée à l'URL XML. Flash accède à l'URL via HTTP GET et s'attend à recevoir des données XML valides.

Une valeur « `send/receive` » signifie que Flash envoie les données XML via HTTP POST et s'attend à recevoir des données XML valides.

Si le paramètre `direction` est `null` ou non reconnu, la valeur par défaut est « `send/receive` ».

ignoreWhite Valeur booléenne : la définition par défaut est `false`. Lorsque ce paramètre est défini sur `true`, les nœuds de texte qui ne contiennent que des espaces vierges sont ignorés au cours de l'analyse. Les nœuds de texte qui contiennent un espace vierge avant ou après leur nom ne sont pas affectés.

multipleSimultaneousAllowed Valeur booléenne : lorsqu'elle est définie sur `true`, elle permet de lancer une opération `trigger()` alors qu'une autre opération `trigger()` est déjà en cours. Lorsque vous lancez simultanément plusieurs opérations `trigger()`, celles-ci ne se termineront pas nécessairement dans l'ordre dans lequel elles ont été appelées. Flash Player peut aussi limiter le nombre d'opérations réseau simultanées. Cette limite dépend de la version et de la plate-forme. Lorsque le paramètre est défini sur `false`, vous ne pouvez pas lancer d'opération `trigger()` si une opération de ce type est en cours.

suppressInvalidCalls Valeur booléenne : lorsque cette valeur est définie sur `true`, elle supprime l'opération `trigger()` si les paramètres de données ne sont pas valides. Lorsque le paramètre `suppressInvalidCall` est défini sur `false`, l'opération `trigger()` s'exécute et utilise des données non valides si nécessaire.

Flux de travail courant du composant XMLConnector

La procédure suivante décrit le flux de travail classique pour le composant XMLConnector.

Pour utiliser un composant XMLConnector :

1. Vérifiez que vos paramètres de publication sont bien définis sur ActionScript 2.0.
2. Ajoutez une occurrence du composant XMLConnector à l'application et donnez-lui un nom d'occurrence.
3. Utilisez l'onglet Paramètres de l'inspecteur des composants pour entrer l'URL de la source de données XML externe à laquelle vous souhaitez accéder.
4. Utilisez l'onglet Schéma de l'inspecteur des composants pour spécifier le schéma du document XML.

REMARQUE	Vous pouvez utiliser le bouton Importer un exemple de schéma pour automatiser ce processus.

5. Utilisez l'onglet Liaisons de l'inspecteur de composants pour lier les éléments de données (`params` et `results`) du document XML aux propriétés des composants visuels de l'application.

   Par exemple, si vous vous connectez à un document XML contenant des données météo, vous pouvez associer les éléments de données Lieu et Température aux composants Label de votre application. Le nom et la température de la ville spécifiée apparaîtront lors de l'exécution.
6. Ajoutez un déclencheur pour lancer l'opération de liaison des données de l'une des façons suivantes :
   - Associez le comportement Déclencher une source de données à un bouton.
   - Ajoutez votre ActionScript pour appeler la méthode `trigger()` sur le composant XMLConnector.
   - Créez une liaison entre un paramètre XML et un bouton de l'interface utilisateur et définissez sa propriété Type sur AutoTrigger. Pour plus d'informations, reportez-vous à *Types de schéma* dans *Utilisation de Flash*.

Pour voir un exemple détaillé de connexion et d'affichage de code XML en utilisant le composant XMLConnector, reportez-vous à Didacticiel XML : Feuille de présence dans les didacticiels d'intégration de données, à l'adresse www.adobe.com/go/data_integration_fr.

Classe XMLConnector

Héritage RPCCall > XMLConnector

Nom de classe ActionScript mx.data.components.XMLConnector

La classe XMLConnector vous permet d'envoyer ou de recevoir des fichiers XML à l'aide de HTTP. Vous pouvez utiliser ActionScript pour lier d'autres composants à une source de données qui renvoie des données XML et permettre ainsi une communication entre les composants.

Méthodes de la classe XMLConnector

Le tableau suivant présente la méthode de la classe XMLConnector.

Méthode	Description
`XMLConnector.trigger()`	Lance un appel de procédure à distance.

Propriétés de la classe XMLConnector

Le tableau suivant répertorie les propriétés de la classe XMLConnector.

Propriété	Description
`XMLConnector.direction`	Indique si les données sont envoyées, reçues ou les deux.
`XMLConnector.ignoreWhite`	Indiquez si les nœuds de texte qui ne contiennent que des espaces vierges sont supprimés au cours de l'analyse.
`XMLConnector.multipleSimultaneousAllowed`	Indique si plusieurs appels peuvent être effectués simultanément.
`XMLConnector.params`	Spécifie les données qui sont envoyées au serveur lors de l'exécution de la prochaine opération `trigger()`.
`XMLConnector.results`	Identifie les données reçues du serveur suite à l'opération `trigger()`.
`XMLConnector.suppressInvalidCalls`	Indique si un appel doit être supprimé lorsque ses paramètres ne sont pas valides.
`XMLConnector.URL`	URL utilisé par le composant dans les opérations HTTP.

Récapitulatif des événements de la classe XMLConnector

Le tableau suivant répertorie les événements de la classe XMLConnector.

Evénement	Description
`XMLConnector.result`	Diffusé lorsqu'un appel de procédure à distance s'achève avec succès.
`XMLConnector.send`	Diffusé lorsque la méthode `trigger()` est en cours, après la collecte des paramètres, mais avant la validation des données et le lancement de l'appel de procédure à distance.
`XMLConnector.status`	Diffusé lorsqu'un appel de procédure à distance est lancé, pour informer l'utilisateur de l'état de l'opération.

XMLConnector.direction

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

componentInstance.direction

Description

Propriété : indique si les données sont envoyées, reçues ou les deux. Les valeurs sont les suivantes :

- `send` Les données XML de la propriété `params` sont envoyées via la méthode `POST` à l'URL du document XML. Toutes les données renvoyées sont ignorées. La propriété `results` n'est pas définie et il n'y a aucun événement `result`.

REMARQUE

Les propriétés `params` et `results` et l'événement `result` sont hérités de l'API du composant RPC.

- `receive` Aucune donnée `params` n'est envoyée à l'URL. Vous pouvez accéder à l'URL du document XML via HTTP `GET`, et l'URL doit envoyer des données XML valides.
- `send/receive` Les données `params` sont envoyées vers l'URL qui doit envoyer des données XML valides.

Exemple

L'exemple suivant définit la direction sur `receive` pour le document `mysettings.xml` :

```
myXMLConnector.direction = "receive";
myXMLConnector.URL = "mysettings.xml";
myXMLConnector.trigger();
```

XMLConnector.ignoreWhite

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`componentInstance``.ignoreWhite`

Description

Propriété : une valeur booléenne. Lorsque ce paramètre est défini sur `true`, les nœuds de texte qui ne contiennent que des espaces vierges sont ignorés au cours de l'analyse. Les nœuds de texte qui contiennent un espace vierge avant ou après leur nom ne sont pas affectés.
Le paramètre par défaut est `false`.

Exemple

Le code suivant définit la propriété `ignoreWhite` sur `true` :

```
myXMLConnector.ignoreWhite = true;
```

XMLConnector.multipleSimultaneousAllowed

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

componentInstance.multipleSimultaneousAllowed

Description

Propriété : indique si plusieurs appels peuvent être effectués simultanément. Si cette propriété est définie sur `false`, la méthode `XMLConnector.trigger()` effectue un appel si un autre appel est déjà en cours. Un événement `status` est émis, avec le code `CallAlreadyInProgress`. Si cette propriété est définie sur `true`, l'appel a lieu.

Lorsque plusieurs appels ont lieu simultanément, il n'est pas garanti qu'ils se termineront dans l'ordre dans lequel ils ont été déclenchés. De plus, le navigateur et/ou le système d'exploitation peut définir des limites sur le nombre d'opérations réseau simultanées. La limite la plus probable que vous risquez de rencontrer est la suivante : le navigateur définit un nombre maximum d'URL pouvant être téléchargées simultanément. Ceci est souvent configurable dans un navigateur. Néanmoins, même dans ce cas, le navigateur doit mettre les flux continus en file d'attente sans interférer avec le comportement prévu de l'application Flash.

Exemple

Cet exemple récupère un fichier XML distant en utilisant le composant XMLConnector en définissant la propriété `direction` sur `receive`. Faites glisser un composant XMLConnector dans votre bibliothèque, puis entrez le code suivant sur l'image 1 du scénario :

```
import mx.data.components.XMLConnector;
var xmlListener:Object = new Object();
xmlListener.result = function(evt:Object) {
  trace("results:");
  trace(evt.target.results);
  trace("");
};
xmlListener.status = function(evt:Object) {
  trace("status::"+evt.code);
};
var myXMLConnector:XMLConnector = new XMLConnector();
myXMLConnector.addEventListener("result", xmlListener);
```

```
myXMLConnector.addEventListener("status", xmlListener);
myXMLConnector.direction = "receive";
myXMLConnector.URL = "http://www.flash-mx.com/mm/tips/tips.xml";
myXMLConnector.multipleSimultaneousAllowed = false;
myXMLConnector.suppressInvalidCalls = true;
myXMLConnector.trigger();
myXMLConnector.trigger();
myXMLConnector.trigger();
```

Cet exemple spécifie l'URL du fichier XML et définit `multipleSimultaneousAllowed` sur `false`. Il déclenche trois fois l'occurrence XMLConnector. La méthode `status` de l'écouteur d'événement affiche le code d'erreur `CallAlreadyInProgress` deux fois dans le panneau Sortie. La première tentative est envoyée de Flash au serveur. Lorsque le premier déclencheur reçoit un résultat, l'événement `result` est diffusé et le paquet XML que vous recevez est affiché dans le panneau Sortie.

XMLConnector.params

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`componentInstance``.params`

Description

Propriété : détermine les données qui seront envoyées au serveur lors de l'exécution de la prochaine opération `trigger()`. Chaque composant RPC définit la manière dont les données sont utilisées et les types valides.

Exemple

L'exemple suivant définit les paramètres `name` et `city` pour `myXMLConnector` :

```
myXMLConnector.params = new XML("<mydoc><name>Bob</name><city>Oakland</
  city></mydoc>");
```

XMLConnector.result

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

componentInstance.addEventListener("result", *myListenerObject*)

Description

Evénement : diffusé lorsqu'un appel de procédure à distance s'achève avec succès.

Le paramètre du gestionnaire d'événements est un objet comportant les champs suivants :

- `type` : la chaîne « `result` »
- `target` : une référence à l'objet qui a provoqué l'événement (un composant WebServiceConnector, par exemple).

Vous pouvez récupérer la valeur réelle du résultat à l'aide de la propriété `results`.

Exemple

L'exemple suivant définit une fonction `res` pour l'événement `result` et affecte la fonction au gestionnaire d'événements `addEventListener` :

```
var res = function (ev) {
trace(ev.target.results);
};
xcon.addEventListener("result", res);
```

XMLConnector.results

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

componentInstance.results

Description

Propriété : identifie les données reçues en provenance du serveur suite à une opération `trigger()`. Chaque composant RPC définit la manière dont les données sont récupérées et les types valides. Ces données apparaissent lorsque l'opération RPC a réussi, comme indiqué par l'événement `result`. Elles sont disponibles jusqu'à la purge du composant ou jusqu'à l'opération RPC suivante.

Les données renvoyées peuvent être très volumineuses. Il existe deux manières de les gérer :

- Sélectionnez un clip, un scénario ou un écran approprié en tant que parent du composant RPC. Lorsque le parent est détruit, la mémoire de stockage de ce composant peut être utilisée pour collecter divers éléments.
- Vous pouvez affecter à tout moment la valeur `null` à cette propriété à l'aide d'ActionScript.

Exemple

L'exemple suivant présente la propriété `results` pour `myXMLConnector` :

```
trace(myXMLConnector.results);
```

XMLConnector.send

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

```
componentInstance.addEventListener("send", myListenerObject)
```

Description

Evénement : diffusé lorsque l'opération `trigger()` est en cours, après la collecte des paramètres, mais avant la validation des données et le lancement de l'appel de procédure distante. Cet événement permet d'insérer du code destiné à modifier les paramètres avant l'appel.

Le paramètre du gestionnaire d'événements est un objet comportant les champs suivants :

- `type` : la chaîne « `send` »
- `target` : une référence à l'objet qui a provoqué l'événement (un composant WebServiceConnector, par exemple).

Vous pouvez récupérer ou modifier les valeurs réelles des paramètres à l'aide de la propriété `params`.

Exemple

L'exemple suivant définit une fonction `sendFunction` pour l'événement `send` et affecte la fonction au gestionnaire d'événements `addEventListener` :

```
var sendFunction = function (sendEnv) {
sendEnv.target.params = [newParam_txt.text];
};
xcon.addEventListener("send", sendFunction);
```

XMLConnector.status

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

componentInstance`.addEventListener("status", `*myListenerObject*`)`

Description

Evénement : diffusé lorsqu'un appel de procédure à distance est lancé, pour informer l'utilisateur de l'état de l'opération.

Le paramètre du gestionnaire d'événements est un objet comportant les champs suivants :

- `type` : la chaîne « `status` »
- `target` : une référence à l'objet qui a provoqué l'événement (un composant WebServiceConnector, par exemple).
- `code` : une chaîne donnant le nom de la condition spécifique qui s'est produite
- `data` : un objet dont le contenu dépend du code

Le champ de code pour l'événement status est défini sur Fault si des problèmes surviennent lors de l'appel, comme suit :

Code	Données	Description
Fault	{faultcode: code, faultstring: string, detail: detail, element: element, faultactor: actor}	Cet événement est provoqué si d'autres problèmes surviennent lors du traitement de l'appel. Les données constituent un objet SOAPFault. Une fois que cet événement s'est produit, l'appel tenté est considéré comme terminé ; aucun événement `result` ou `send` ne se produit.

Les erreurs suivantes peuvent se produire avec l'événement `status` :

Code d'erreur	Chaîne	Remarques
`XMLConnector.Not.XML`	params is not an XML object	La valeur `params` doit être un objet XML d'ActionScript.
`XMLConnector.Parse.Error`	params had XML parsing error NN	La propriété `status` de l'objet XML `params` a une valeur NN différente de zéro. Pour visualiser les erreurs NN possibles, reportez-vous à `XML.status` dans le *Guide de référence du langage ActionScript 2.0*.
`XMLConnector.No.Data.Received`	no data was received from the server	En raison des limitations propres au navigateur, ce message peut signifier que (a) l'URL du serveur n'était pas valide, ne répondait pas ou a renvoyé un code d'erreur HTTP ; ou (b) que la demande du serveur a réussi mais que la réponse contenait 0 octets de données. Pour contourner cette restriction, créez votre application de façon à ce que le serveur ne renvoie jamais 0 octets de données. Si vous recevez le code d'erreur `XMLConnector.No.Data.Received`, vous savez qu'une erreur serveur s'est produite et vous pouvez en informer l'utilisateur.

Code d'erreur	Chaîne	Remarques
`XMLConnector.Results.Parse.Error`	received data had an XML parsing error NN	Le code XML reçu n'était pas valide, comme indiqué par le programme d'analyse XML intégré de Flash Player. Pour visualiser les erreurs NN possibles, reportez-vous à `XML.status` dans le *Guide de référence du langage ActionScript 2.0*.
`XMLConnector.Params.Missing`	Direction is 'send' or 'send/receive', but params are null.	

Exemple

L'exemple suivant définit une fonction `statusFunction` pour l'événement `status` et affecte la fonction au gestionnaire d'événements `addEventListener` :

```
var statusFunction = function (stat) {
trace(stat.code);
trace(stat.data.faultcode);
trace(stat.data.faultstring);
};
xcon.addEventListener("status", statusFunction);
```

XMLConnector.suppressInvalidCalls

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`componentInstance``.suppressInvalidCalls`

Description

Propriété : indique si un appel doit être supprimé lorsque ses paramètres ne sont pas valides. Si cette propriété est définie sur `true`, la méthode `trigger()` n'effectue aucun appel si les paramètres liés ne réalisent pas la validation. Un événement `status` est émis, avec le code `InvalidParams`. Si cette propriété est définie sur `false`, l'appel a lieu, en utilisant les données non valides, comme nécessaire.

Exemple

Cet exemple affiche une erreur car les paramètres nécessaires ne sont pas transmis. Faites glisser un composant XMLConnector dans votre bibliothèque, puis entrez le code suivant sur l'image 1 du scénario :

```
import mx.data.components.XMLConnector;
var xmlListener:Object = new Object();
xmlListener.result = function(evt:Object) {
  trace("results:");
  trace(evt.target.results);
  trace("");
};
xmlListener.status = function(evt:Object) {
  switch (evt.code) {
  case 'Fault' :
    trace("ERROR! ["+evt.data.faultcode+"]");
    trace("\t"+evt.data.faultstring);
    break;
  case 'InvalidParams' :
    trace("ERROR! ["+evt.code+"]");
    break;
  }
};
var myXMLConnector:XMLConnector = new XMLConnector();
myXMLConnector.addEventListener("result", xmlListener);
myXMLConnector.addEventListener("status", xmlListener);
myXMLConnector.direction = "send/receive";
myXMLConnector.URL = "http://www.flash-mx.com/mm/login_xml.cfm";
myXMLConnector.multipleSimultaneousAllowed = false;
myXMLConnector.suppressInvalidCalls = false;
// myXMLConnector.params = new XML("<login username='Mort'
// password='Guacamole' />");
myXMLConnector.trigger();
```

Supprimez les commentaires de la deuxième à la dernière ligne de code pour que le fragment de code fonctionne correctement.

XMLConnector.trigger()

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

componentInstance.trigger()

Description

Méthode : lance un appel de procédure à distance par le composant XMLConnector. Ceci peut se faire par l'obtention ou l'envoi au fichier XML spécifié. Si l'opération réussit, les résultats apparaissent dans la propriété `results` du composant RPC.

La méthode `trigger()` effectue les opérations suivantes :

1. Si des données sont liées à la propriété `params`, la méthode exécute toutes les liaisons afin de garantir que les données disponibles sont à jour. Cela provoque également la validation des données.
2. Si les données ne sont pas valides alors que le paramètre `suppressInvalidCalls` est défini sur `true`, l'opération est interrompue.
3. Si l'opération continue, l'événement `send` se produit.
4. L'appel distant est lancé à l'aide de la méthode de connexion indiquée (HTTP, par exemple).

Exemple

Cet exemple récupère un fichier XML distant en utilisant le composant XMLConnector en définissant la propriété `direction` sur `receive`. Faites glisser un composant XMLConnector dans votre bibliothèque, puis entrez le code suivant sur l'image 1 du scénario :

```
import mx.data.components.XMLConnector;
var xmlListener:Object = new Object();
xmlListener.result = function(evt:Object) {
  trace("results:");
  trace(evt.target.results);
  trace("");
};
xmlListener.status = function(evt:Object) {
  trace("status::"+evt.code);
};
var myXMLConnector:XMLConnector = new XMLConnector();
```

```
myXMLConnector.addEventListener("result", xmlListener);
myXMLConnector.addEventListener("status", xmlListener);
myXMLConnector.direction = "receive";
myXMLConnector.URL = "http://www.flash-mx.com/mm/tips/tips.xml";
myXMLConnector.multipleSimultaneousAllowed = false;
myXMLConnector.suppressInvalidCalls = true;
myXMLConnector.trigger();
myXMLConnector.trigger();
myXMLConnector.trigger();
```

Ce code spécifie l'URL du fichier XML et définit `multipleSimultaneousAllowed` sur `false`. Il déclenche trois fois l'occurrence XMLConnector. La méthode `status` de l'écouteur d'événement affiche le code d'erreur `CallAlreadyInProgress` deux fois dans le panneau Sortie. La première tentative est envoyée de Flash au serveur. Lorsque le premier déclencheur reçoit un résultat, l'événement `result` est diffusé et le paquet XML que vous recevez est affiché dans le panneau Sortie.

XMLConnector.URL

Disponibilité

Flash Player 6 (6.0.79.0).

Edition

Flash MX Professional 2004.

Utilisation

`componentInstance``.URL`

Description

Propriété : l'URL que ce composant utilise lors de l'exécution des opérations HTTP. Il peut s'agir d'une URL absolue ou relative. L'URL est soumise à toutes les fonctionnalités de sécurité standard de Flash Player (pour plus d'informations sur ces fonctionnalités, reportez-vous à « Fonctionnement de la sécurité » dans le guide *Formation à ActionScript 2.0 dans Adobe Flash*).

Exemple

Cet exemple récupère un fichier XML distant en utilisant le composant XMLConnector en définissant la propriété `direction` sur `receive`. Faites glisser un composant XMLConnector dans votre bibliothèque, puis entrez le code suivant sur l'image 1 du scénario :

```
import mx.data.components.XMLConnector;
var xmlListener:Object = new Object();
xmlListener.result = function(evt:Object) {
```

```
  trace("results:");
  trace(evt.target.results);
  trace("");
};
xmlListener.status = function(evt:Object) {
  trace("status::"+evt.code);
};
var myXMLConnector:XMLConnector = new XMLConnector();
myXMLConnector.addEventListener("result", xmlListener);
myXMLConnector.addEventListener("status", xmlListener);
myXMLConnector.direction = "receive";
myXMLConnector.URL = "http://www.flash-mx.com/mm/tips/tips.xml";
myXMLConnector.multipleSimultaneousAllowed = false;
myXMLConnector.suppressInvalidCalls = true;
myXMLConnector.trigger();
myXMLConnector.trigger();
myXMLConnector.trigger();
```

Ce code spécifie l'URL du fichier XML et définit `multipleSimultaneousAllowed` sur `false`. Il déclenche trois fois l'occurrence XMLConnector. La méthode `status` de l'écouteur d'événement affiche le code d'erreur `CallAlreadyInProgress` deux fois dans le panneau Sortie. La première tentative est envoyée de Flash au serveur. Lorsque le premier déclencheur reçoit un résultat, l'événement `result` est diffusé et le paquet XML que vous recevez est affiché dans le panneau Sortie.

CHAPITRE 60

Classe XPathAPI

60

Nom de classe ActionScript mx.xpath.XPathAPI

La classe XPathAPI vous permet d'effectuer des recherches XPath simples dans Adobe Flash. Cette procédure peut être très utile pour rechercher des paquets XML en fonction de noms de nœud ou de valeurs d'attribut. En d'autres termes, vous pouvez trouver rapidement des nœuds et des attributs dans un document XML en utilisant les méthodes XpathAPI.

REMARQUE

La classe XPathAPI est prise en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Pour utiliser les recherches XPath dans Flash, ajoutez d'abord la classe DataBindingClass (si ce n'est déjà fait) pour inclure la classe XPathAPI dans la bibliothèque Flash. Si vous avez déjà configuré des liaisons, cette classe a peut-être été incluse automatiquement. Dans le cas contraire, vous devez la sélectionner dans les bibliothèques communes (Fenêtre > Bibliothèques communes > Classes). A partir du panneau Bibliothèque Classes.fla, vous pouvez faire glisser simplement une copie du composant DataBindingClasses dans la bibliothèque de votre document Flash en cours. A présent, vous pouvez importer la classe en tapant `import mx.xpath.XPathAPI` ou en utilisant son nom complet pour accéder à ses méthodes en les faisant précéder de `mx.xpath.XPathAPI.`*`nom_méthode`*.

Pour plus d'informations sur cette classe, visitez le Centre de ressources et de documentation de Flash à l'adresse suivante www.adobe.com/go/xpathapi_fr.

CHAPITRE 61

Composant XUpdateResolver

61

Les composants Resolver s'utilisent en association avec le composant DataSet (partie de la fonctionnalité de gestion des données dans l'architecture de données Flash) pour enregistrer des changements dans une source de données externe.

REMARQUE	Les composants resolver sont pris en charge uniquement si vous travaillez dans un document spécifiant ActionScript 2.0 dans ses paramètres de publication.

Ils prennent un objet `DataSet.deltaPacket` et le convertissent en un paquet de mise à jour dans un format approprié au type de composant Resolver. Le paquet de mise à jour peut ensuite être transmis à la source de données externe par l'un des composants Connector. Les composants Resolver ne sont pas visibles lors de l'exécution.

Pour plus d'informations sur la gestion des données de Flash via le composant DataSet, reportez-vous à *Gestion des données* dans *Utilisation de Flash*.

XUpdate est un standard pour la description des modifications apportées à un document XML ; il est pris en charge par diverses bases de données XML, par exemple Xindice et XHive. Le composant XUpdateResolver traduit les modifications apportées à un composant DataSet en instructions XUpdate. Les mises à jour à partir du composant XUpdateResolver sont envoyées sous la forme d'un paquet de données XUpdate, qui est communiqué à la base de données ou au serveur par l'intermédiaire d'un objet de connexion. Le composant XUpdateResolver reçoit un paquet delta d'un composant DataSet, envoie son propre paquet de mise à jour à un connecteur, reçoit les erreurs du serveur de la connexion et les communique en retour au composant DataSet. Tous ces processus utilisent les propriétés de liaison.

Pour plus d'informations sur le modèle de spécification du langage XUpdate, consultez la page http://xmldb-org.sourceforge.net/xupdate/xupdate-wd.html. Pour plus d'informations sur l'architecture de données Flash, reportez-vous à *Résolution des données* dans *Utilisation de Flash* ; pour plus d'informations sur la résolution de données XML, reportez-vous à *Résolution des données XML avec le composant XUpdateResolver* dans *Utilisation de Flash*.

CONSEIL

Vous pouvez également utiliser le composant XUpdateResolver pour envoyer des mises à jour de données à toute source de données externe pouvant analyser le langage XUpdate (par exemple, une page ASP, un servlet Java ou un composant ColdFusion).

Utilisation du composant XUpdateResolver

Vous utilisez le composant XUpdateResolver uniquement lorsque votre application Flash contient un composant DataSet et doit renvoyer une mise à jour à la source de données.

Le composant XUpdateResolver communique avec le composant DataSet à l'aide du codeur DataSetDeltaToXUpdateDelta. Ce codeur crée des instructions XPath qui identifient des nœuds de façon unique dans un fichier XML en fonction des informations contenues dans le paquet delta du composant DataSet. Ces informations sont utilisées par le composant XUpdateResolver pour générer des instructions XUpdate. Pour plus d'informations sur le codeur DataSetDeltaToXUpdateDelta, reportez-vous à la section *Encodeurs de schémas* dans *Utilisation de Flash*.

Pour plus d'informations sur l'utilisation du composant XUpdateResolver, reportez-vous à *Résolution des données* dans *Utilisation de Flash*.

Paramètre du composant XUpdateResolver

Le composant XUpdateResolver dispose d'un paramètre de création booléen `includeDeltaPacketInfo`. Lorsque ce paramètre est défini sur `true`, le paquet de mise à jour inclut des informations supplémentaires qui peuvent être utilisées par une source de données externe pour générer des résultats pouvant être renvoyés vers votre application.
Ces informations contiennent un ID d'opération et de transaction unique utilisé en interne par le jeu de données.

REMARQUE	Les autres informations incluses dans le paquet de mise à jour invalident le XUpdate. Vous ne choisissez d'ajouter ces informations que si vous souhaitez les stocker dans un objet serveur et les utiliser pour générer un paquet de résultats. Dans ce scénario, votre objet serveur extrait les informations du paquet de mise à jour pour ses propres besoins, puis transmet le XUpdate (désormais valide) à la base de données.

L'exemple suivant décrit un paquet de mise à jour XML lorsque le paramètre `includeDeltaPacketInfo` est défini sur `false` :

```
<xupdate:modifications version="1.0" xmlns:xupdate="http://www.xmldb.org/
  xupdate">
    <xupdate:remove select="/datapacket/row[@id='100']"/>
</xupdate:modifications>
```

L'exemple suivant décrit un paquet de mise à jour XML lorsque le paramètre `includeDeltaPacketInfo` est défini sur `true` :

```
<xupdate:modifications version="1.0" xmlns:xupdate="http://www.xmldb.org/
  xupdate"
     transId="46386292065:Wed Jun 25 15:52:34 GMT-0700 2003">
    <xupdate:remove select="/datapacket/row[@id='100']" opId="0123456789"/>
</xupdate:modifications>
```

Flux de travaux courant pour le composant XUpdateResolver

La procédure suivante décrit le flux de travaux courant pour le composant XUpdateResolver.

Pour utiliser un composant XUpdateResolver :

1. Vérifiez que vos paramètres de publication sont bien définis sur ActionScript 2.0.
2. Ajoutez deux occurrences du composant XMLConnector, une occurrence du composant DataSet et une occurrence du composant XUpdateResolver à votre application et donnez-leur des noms d'occurrence.

3. Sélectionnez le premier composant XMLConnector, puis utilisez l'onglet Paramètres de l'inspecteur de composants pour entrer l'URL de la source de données XML externe à laquelle vous souhaitez accéder.
4. Le composant XMLConnector étant toujours sélectionné, cliquez sur l'onglet Schéma de l'inspecteur de composants et importez un exemple de fichier XML pour générer votre schéma.

REMARQUE	Vous devrez peut-être créer un schéma virtuel pour votre fichier XML si vous souhaitez accéder à un sous-élément du tableau que vous liez au jeu de données. Pour plus d'informations, reportez-vous à *Schémas virtuels* dans *Utilisation de Flash*.

5. Utilisez l'onglet Liaisons de l'inspecteur de composants pour lier un tableau du composant XMLConnector à la propriété `dataProvider` du composant DataSet.
6. Sélectionnez le composant DataSet et utilisez l'onglet Schéma de l'inspecteur des composants pour créer les champs DataSet qui seront liés aux champs de l'objet dans le tableau.
7. Utilisez l'onglet Liaisons de l'inspecteur des composants pour lier des éléments de données (champs DataSet) aux composants visuels de votre application.
8. Sélectionnez l'onglet Schéma du composant XUpdateResolver. Sélectionnez la propriété de composant `deltaPacket`, puis utilisez le panneau Attributs du schéma pour définir la propriété `encoder` sur le codeur DataSetDeltaToXUpdateDelta.
9. Sélectionnez Options du codeur, puis entrez la valeur `rowNodeKey` qui identifie de façon unique le nœud de lignes du fichier XML.

REMARQUE	La valeur `rowNodeKey` combine une instruction XPath avec un paramètre de champ pour définir la façon dont les instructions XPath uniques doivent être générées pour les données de mise à jour contenues dans le paquet delta. Pour plus d'informations sur le codeur DataSetDeltaToXUpdateDelta, reportez-vous à *Encodeurs de schémas* dans *Utilisation de Flash*.

10. Cliquez sur l'onglet Liaisons, puis créez une liaison entre la propriété `deltaPacket` du composant XUpdateResolver et la propriété `deltaPacket` du composant DataSet.

11. Créez une autre liaison entre la propriété `xupdatePacket` et le deuxième composant XMLConnector pour renvoyer les données vers la source de données externe.

REMARQUE	La propriété `xupdatePacket` contient le paquet delta formaté (instructions XUpdate) qui sera envoyé au serveur.

12. Ajoutez un déclencheur pour lancer l'opération de liaison des données : utilisez le comportement Déclencher une source de données associé à un bouton ou ajoutez des instructions ActionScript.

Outre ces étapes, vous pouvez également créer des liaisons pour appliquer le paquet de résultats renvoyé à partir du serveur au jeu de données via le composant XUpdateResolver.

Pour voir un exemple détaillé d'une résolution de données sur une source de données externe en utilisant le composant XUpdate, reportez-vous à *Mise à jour de la feuille de présence* que vous trouverez dans les didacticiels sur l'intégration des données à l'adresse suivante www.adobe.com/go/data_integration_fr.

Classe XUpdateResolver

Héritage MovieClip > XUpdateResolver

Nom de classe ActionScript mx.data.components.XUpdateResolver

Les propriétés et événements de la classe XUpdateResolver vous permettent d'utiliser le composant DataSet pour enregistrer les modifications apportées aux sources de données externes.

Propriétés de la classe XUpdateResolver

Le tableau suivant répertorie les propriétés de la classe XUpdateResolver.

Propriété	Description
`XUpdateResolver.deltaPacket`	Contient une description des modifications apportées au composant DataSet. La propriété `deltaPacket` du composant DataSet doit être liée à cette propriété afin que la liaison la copie lors de l'appel à la méthode `DataSet.applyUpdates()` et que le composant Resolver crée le paquet XUpdate.
`XUpdateResolver.includeDeltaPacketInfo`	Comprend des informations supplémentaires du paquet delta dans les attributs des nœuds XUpdate.
`XUpdateResolver.updateResults`	Décrit les résultats d'une mise à jour.
`XUpdateResolver.xupdatePacket`	Contient la traduction XUpdate des modifications vers le composant DataSet.

Evénements de la classe XUpdateResolver

Le tableau suivant répertorie les événements de la classe XUpdateResolver.

Evénement	Description
`XUpdateResolver.beforeApplyUpdates`	Appelé par le composant Resolver pour apporter des modifications personnalisées immédiatement après la création du paquet XML et juste avant son envoi.
`XUpdateResolver.reconcileResults`	Appelé par le composant Resolver pour comparer deux paquets.

XUpdateResolver.beforeApplyUpdates

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

resolveData.beforeApplyUpdates(*eventObject*)

Paramètres

eventObject Objet événement Resolver ; décrit les personnalisations apportées au paquet XML avant l'envoi de la mise à jour à la base de données par le biais du connecteur. Cet objet événement doit contenir les propriétés suivantes :

Propriété	Description
target	Objet : composant Resolver générant cet événement.
type	Chaîne ; nom de l'événement.
updatePacket	Objet XML : l'objet XML qui sera appliqué.

Renvoie

Aucune.

Description

Evénement : appelé par le composant Resolver pour apporter des modifications personnalisées immédiatement après la création du paquet XML pour un nouveau paquet delta et juste avant l'envoi de ce paquet à l'aide de la liaison de données. Vous pouvez utiliser ce gestionnaire d'événement pour effectuer des modifications personnalisées dans le XML avant d'envoyer les données mises à jour vers un connecteur.

Exemple

L'exemple suivant ajoute les données d'authentification de l'utilisateur au paquet XML :

```
on (beforeApplyUpdates) {
   // Ajout des données d'authentification de l'utilisateur.
   var userInfo = new XML(""+getUserId()+" "+getPassword()+"");
   xupdatePacket.firstChild.appendChild(userInfo);
}
```

XUpdateResolver.deltaPacket

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

resolveData.deltaPacket

Description

Propriété : contient une description des modifications apportées au composant DataSet. Cette propriété est de type `deltaPacket`, reçoit un paquet delta à traduire en paquet XUpdate et génère un paquet delta des résultats du serveur placé dans la propriété `updateResults`. Cette propriété vous permet d'apporter des modifications personnalisées au paquet XML avant l'envoi des données mises à jour au connecteur.

Les messages dans la propriété `updateResults` sont considérés comme des erreurs. Cela signifie qu'un delta contenant des messages est de nouveau ajouté au paquet delta pour être renvoyé à la prochaine transmission de ce paquet au serveur. Vous devez écrire du code qui gère les deltas contenant des messages afin que ces derniers soient présentés à l'utilisateur et modifiés avant d'être ajoutés au paquet delta suivant.

La propriété `deltaPacket` du composant DataSet doit être liée à cette propriété afin que la liaison la copie lors de l'appel à la méthode `DataSet.applyUpdates()` et que le composant Resolver crée le paquet XUpdate.

XUpdateResolver.includeDeltaPacketInfo

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

resolveData.includeDeltaPacketInfo

Description

Propriété : propriété booléenne qui, si elle est définie sur `true`, insère des informations supplémentaires du paquet delta dans les attributs sur les nœuds XUpdate. Ces informations comprennent l'ID de transaction et l'ID d'opération.

Pour consulter un exemple de XML obtenu, reportez-vous à « Paramètre du composant XUpdateResolver », à la page 1567.

XUpdateResolver.reconcileResults

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

resolveData.reconcileResults(*eventObject*)

Paramètres

eventObject Objet ResolverEvent ; décrit l'objet événement utilisé pour comparer deux paquets de mise à jour. Cet objet événement doit contenir les propriétés suivantes :

Propriété	Description
`target`	Objet ; composant Resolver générant cet événement.
`type`	Chaîne ; nom de l'événement.

Renvoie

Aucun.

Description

Evénement : appelé par le composant Resolver pour comparer deux paquets. Utilisez ce rappel pour insérer tout code après réception des résultats du serveur et juste avant la transmission, par la liaison de données, du paquet delta contenant les résultats de l'opération. Il s'agit d'un bon emplacement pour insérer du code qui gère les messages en provenance du serveur.

Exemple

L'exemple suivant reconstitue deux paquets de mise à jour et efface les mises à jour en cas de succès :

```
on (reconcileResults) {
      // Examen des résultats.
      if(examine(updateResults))
        myDataSet.purgeUpdates();
      else
       displayErrors(results);
}
```

XUpdateResolver.updateResults

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

resolveData.updateResults

Description

Propriété : propriété de type `deltaPacket` qui contient les résultats d'une mise à jour renvoyés à partir du serveur à l'aide d'un connecteur. Utilisez cette propriété pour transmettre les erreurs et les données mises à jour à partir du serveur vers un composant DataSet ; par exemple, lorsque le serveur affecte de nouveaux ID pour un champ auto-affecté. Liez cette propriété à la propriété `results` d'un connecteur afin que celui-ci puisse recevoir les résultats d'une mise à jour et les retransmettre au composant DataSet.

Les messages dans la propriété `updateResults` sont considérés comme des erreurs.
Cela signifie qu'un delta contenant des messages est de nouveau ajouté au paquet delta pour être renvoyé à la prochaine transmission de ce paquet au serveur. Vous devez écrire du code qui gère les deltas contenant des messages afin que ces derniers soient présentés à l'utilisateur et modifiés avant d'être ajoutés au paquet delta suivant.

XUpdateResolver.xupdatePacket

Disponibilité

Flash Player 7.

Edition

Flash MX Professional 2004.

Utilisation

resolveData.xupdatePacket

Description

Propriété : propriété de type `xml` qui contient la traduction XUpdate des modifications apportées au composant DataSet. Effectuez la liaison de cette propriété avec la propriété du composant connecteur qui retransmet le paquet de mise à jour traduit des modifications vers le composant DataSet.

Index

A

B

C

D

E

F

I

J

L

M

O

P

S

T

U

X