FORMATION À ACTIONSCRIPT™ 2.0 DANS FLASH®

Adobe®

Formation à ActionScript 2.0 dans Flash®

Table des matières

Introduction

Adobe Flash CS3 Professional est l'outil préféré des professionnels pour la création de contenu Web percutant. Le langage ActionScript permet de rendre les applications Flash plus interactives, qu'il s'agisse de simples fichiers SWF animés ou d'applications Internet plus complexes. Vous pouvez très bien utiliser Flash sans ActionScript. Toutefois, si l'interactivité des utilisateurs est un critère important ou si vous souhaitez utiliser des objets autres que ceux intégrés dans Flash (boutons et clips) ou créer des applications plus complexes à partir de fichiers SWF, il est probable que vous aurez recours à ActionScript.

Pour plus d'informations, voir les sections suivantes :

Public visé

Ce manuel suppose que vous avez déjà installé Flash et savez comment utiliser l'interface utilisateur.Vous devez également savoir comment placer des objets sur la scène et les manipuler dans l'environnement de programmation de Flash. Si vous avez déjà utilisé un langage de programmation, vous vous sentirez en terrain connu avec ActionScript. Cependant, si vous êtes novice en programmation, les bases d'ActionScript sont faciles à apprendre. Commencez par les commandes de base, puis abordez progressivement les éléments plus complexes. Vous pouvez donc accroître l'interactivité de vos fichiers sans avoir à apprendre (ou à écrire) beaucoup de code.

Configuration système requise

La configuration système requise par ActionScript 2.0 est identique à celle de Flash.

Flash CS3 Professional introduit ActionScript 3.0. Flash Player 9 et ActionScript 3.0 sont les paramètres de publication par défaut de Flash. Ce manuel fournit des informations sur l'utilisation d'ActionScript 2.0 avec Flash. Vous devez modifier les paramètres de publication de vos fichiers Flash sur Flash Player 9 et ActionScript 2.0. Si vous ne modifiez pas ces paramètres, les explications et les exemples de code qui figurent dans la documentation risquent de ne pas fonctionner correctement. Si vous développez des applications pour les versions précédentes de Flash Player, reportez-vous à l'Annexe D, « Ecriture de scripts destinés à des versions antérieures de Flash Player », à la page 773.

Mise à jour des fichiers XML Flash

Il est essentiel de disposer en permanence des fichiers Flash XML les plus récents. Adobe introduit parfois de nouvelles fonctionnalités dans les versions secondaires de Flash Player. Lorsque ce type de version est disponible, vous devez mettre votre version de Flash à jour. Sinon, le compilateur de Flash risque de générer des erreurs si vous utilisez de nouvelles propriétés ou méthodes non disponibles dans la version de Flash Player livrée avec votre installation Flash.

Par exemple, Flash Player 7 (7.0.19.0) contient une nouvelle méthode pour l'objet System, `System.security.loadPolicyFile`. Pour accéder à cette méthode, vous devez utiliser le programme d'installation des mises à jour de Flash Player. Sinon, le compilateur Flash affiche des erreurs.

N'oubliez pas que vous pouvez installer Player Updater même si ce programme a plusieurs versions d'avance sur votre version de Flash. Ainsi, vous bénéficiez des fichiers XML nécessaires sans rencontrer d'erreurs de compilation lorsque vous publiez des documents destinés à des versions plus anciennes de Flash Player. Certaines nouvelles méthodes ou propriétés sont disponibles pour les anciennes versions. Le fait de disposer des fichiers XML réduit le risque d'erreurs de compilation lorsque vous tentez d'accéder à d'anciennes méthodes ou propriétés.

Présentation de la documentation

Ce manuel passe en revue les principes généraux de la syntaxe ActionScript et explique comment utiliser ce langage pour intervenir sur différents types d'objet. Pour plus d'informations sur la syntaxe et l'utilisation de chaque élément, consultez le *Guide de référence du langage ActionScript 2.0*.

Pour plus d'informations, voir les sections suivantes :

- « Présentation du manuel Formation à ActionScript 2.0 », à la page 11
- « Présentation des fichiers d'exemple », à la page 15
- « Terminologie utilisée dans ce manuel », à la page 13
- « Copie et collage de code », à la page 14

Présentation du manuel Formation à ActionScript 2.0

La liste suivante récapitule le contenu du manuel :

- Chapitre 1, « Nouveautés du langage ActionScript de Flash », décrit les nouvelles fonctions d'ActionScript, les modifications apportées au module de compilation et débogage, ainsi que le nouveau modèle de programmation du langage ActionScript 2.0.
- Chapitre 2, « A propos d'ActionScript », définit le langage ActionScript et explique comment choisir la version d'ActionScript à utiliser.
- Chapitre 3, « Données et types de données », décrit la terminologie et les concepts de base sur les données, les types de données et les variables. Vous aurez recours à ces concepts tout au long du manuel.
- Chapitre 4, « Eléments fondamentaux du langage et de la syntaxe », décrit la terminologie et les concepts de base du langage ActionScript. Vous aurez recours à ces concepts tout au long du manuel.
- Chapitre 5, « Fonctions et méthodes », explique comment écrire différents types de fonctions et de méthodes et comment les utiliser dans votre application.
- Chapitre 6, « Classes », explique comment créer des classes et des objets personnalisés dans ActionScript. Ce chapitre affiche également la liste des classes intégrées dans ActionScript et offre un bref aperçu de leur application pour accéder aux fonctions puissantes du code ActionScript.
- Chapitre 7, « Héritage », décrit l'héritage dans le langage ActionScript et explique comment étendre des classes personnalisées ou intégrées.

- Chapitre 8, « Interfaces », explique comment créer et utiliser les interfaces dans ActionScript.
- Chapitre 9, « Gestion d'événements », décrit les différentes façons de gérer les événements : méthodes de gestionnaire d'événement, écouteurs d'événement et gestionnaires d'événement de bouton et de clip.
- Chapitre 10, « Utilisation des clips », décrit les clips et le code ActionScript qui permet de les contrôler.
- Chapitre 11, « Utilisation du texte et des chaînes », décrit les différentes façons mises à votre disposition pour contrôler le texte et les chaînes dans Flash et comprend des informations sur la mise en forme du texte et de l'anti-aliasing avancé.
- Chapitre 12, « Animation, Filtres et Dessins », explique comment créer des images et des animations à base de code, ajouter des filtres aux objets et dessiner à l'aide d'ActionScript.
- Chapitre 13, « Création d'interactivité avec ActionScript », décrit des méthodes simples de création d'applications plus interactives, ce qui inclut le contrôle de la lecture des fichiers SWF, la création de pointeurs personnalisés et la création de contrôles audio.
- Chapitre 14, « Utilisation des images, du son et de la vidéo », spécifie comment importer des fichiers de supports externes, tels que des images bitmap, des fichiers MP3, des fichiers FLV (Flash Video) et autres fichiers SWF dans vos applications Flash. Ce chapitre explique également comment utiliser la vidéo dans vos applications et comment créer des animations de chargement avec barre de progression.
- Chapitre 15, « Utilisation de données externes », indique comment traiter les données des sources externes en utilisant des scripts côté serveur ou client dans vos applications. Ce chapitre explique comment intégrer des données dans vos applications.
- Chapitre 16, « Fonctionnement de la sécurité », explique comment la sécurité est mise en oeuvre dans Flash Player, en ce qui concerne l'utilisation locale de fichiers SWF sur votre disque dur. Ce chapitre décrit également les problèmes de sécurité inter-domaines et explique comment charger des données à partir de serveur ou à travers des domaines.
- Chapitre 17, « Recommandations et conventions de programmation pour ActionScript 2.0 », détaille les meilleures pratiques d'utilisation de Flash et d'écriture de code ActionScript. Ce chapitre dresse également la liste des conventions de codage normalisées, notamment en matière d'appellation des variables, et d'autres conventions.
- Annexe A, « Messages d'erreur », dresse la liste des messages d'erreur que le compilateur Flash est susceptible de générer.
- Annexe B, « Opérateurs Flash 4 déconseillés », dresse la liste de tous les opérateurs Flash 4 déconseillés et leur associativité.

- Annexe C, « Touches du clavier et valeurs de code correspondantes », répertorie toutes les touches d'un clavier standard et les valeurs de code ASCII correspondantes utilisées pour identifier les touches dans ActionScript.
- Annexe D, « Ecriture de scripts destinés à des versions antérieures de Flash Player », fournit des consignes pour la syntaxe des scripts en fonction de la version de Flash Player ciblée.
- Annexe E, « Programmation orientée objet avec ActionScript 1.0 », fournit des informations sur l'utilisation du modèle d'objet ActionScript 1.0 pour écrire des scripts.

Ce manuel explique comment utiliser le langage ActionScript. Pour plus d'informations sur les éléments de langage, consultez le *Guide de référence du langage ActionScript 2.0*.

Conventions typographiques

Ce manuel utilise les conventions typographiques suivantes :

- `La police de code` indique le code ActionScript.
- `La police de code en gras`, généralement utilisée dans les procédures, signale le code à modifier ou à ajouter au code déjà inséré dans votre fichier FLA. Dans certains cas, elle peut signaler le code à examiner.
- **Le texte en gras** indique des données à saisir dans l'interface utilisateur, telles que le nom d'un fichier ou d'une occurrence.
- *Le texte en italique* indique un terme nouveau défini dans la suite du texte. Dans un chemin de fichier, il peut indiquer une valeur à remplacer (par exemple par le nom d'un répertoire de votre propre disque dur).

Terminologie utilisée dans ce manuel

Ce manuel emploie les termes suivants :

- *Vous* fait référence au programmeur qui développe un script ou une application.
- *L'utilisateur* désigne la personne qui exécute vos scripts et applications.
- La *compilation* correspond à la phase de publication, d'exportation, de test ou de débogage de votre document.
- L'*exécution* représente le moment auquel votre script s'exécute dans Flash Player.

Copie et collage de code

Vous devez faire particulièrement attention aux caractères spéciaux lorsque vous collez des éléments ActionScript du panneau Aide vers votre fichier FLA ou ActionScript. Parmi les caractères spéciaux figurent des guillemets spéciaux (également appelés guillemets anglais ou guillemets typographiques). Ces caractères ne sont pas interprétés par l'éditeur ActionScript. Par conséquent, votre code renvoie une erreur si vous tentez de les compiler dans Flash.

Vous pouvez déterminer que vos guillemets sont des caractères spéciaux si leur code couleur n'est pas correct. En d'autres termes, si toutes vos chaînes ne changent pas de couleur dans l'éditeur de code, vous devez remplacer les caractères spéciaux par des guillemets droits classiques. Si vous tapez un guillemet simple ou double directement dans l'éditeur ActionScript, utilisez toujours un guillemet droit. Le compilateur (lors du test ou de la publication d'un fichier SWF) renvoie une erreur et indique si votre code comporte le mauvais type de caractère (guillemets spéciaux ou guillemets anglais).

REMARQUE	Vous pouvez également rencontrer des guillemets spéciaux lorsque vous collez des éléments ActionScript à partir d'autres emplacements, tels qu'une page Web ou un document Microsoft Word.

Veillez à respecter les sauts de ligne lorsque vous copiez et collez du code. Si vous collez du code à partir de différents emplacements, les sauts de la ligne de code peuvent être situés au mauvais endroit. Assurez-vous que le code couleur de votre syntaxe est correct dans l'éditeur ActionScript si vous pensez que les sauts de ligne peuvent poser problème. Si vous le souhaitez, comparez votre code du panneau Actions à celui du panneau Aide pour voir s'ils correspondent. Essayez d'activer le retour à la ligne dans l'éditeur ActionScript pour vous aider à résoudre l'excédent de sauts de ligne dans votre code (sélectionnez Affichage > Retour à la ligne dans la fenêtre de script ou Retour à la ligne dans le menu déroulant du panneau Actions).

Ressources supplémentaires

En plus de ce manuel sur ActionScript, vous pouvez consulter des manuels sur d'autres sujets liés à Flash, tels que les composants et Adobe Flash Lite. Vous pouvez accéder à chaque manuel dans le panneau Aide (Aide > Aide de Flash), en consultant le Sommaire par défaut. Cliquez sur le bouton Effacer pour faire apparaître chaque manuel disponible. Pour plus d'informations, consultez la section « Documentation se rapportant à d'autres sujets », à la page 17.

Pour plus d'informations sur les autres ressources disponibles, consultez les sections suivantes :

Présentation des fichiers d'exemple

De nombreux fichiers d'exemple basés sur ActionScript sont installés avec Flash. Ces fichiers montrent comment le code fonctionne dans un fichier FLA. Ils constituent généralement un outil d'apprentissage utile. Les chapitres de ce manuel font souvent référence à ces fichiers mais nous vous recommandons également de consulter le dossier de fichiers d'exemple de votre disque dur.

Parmi les fichiers d'exemple figurent des fichiers d'application FLA qui utilisent des fonctionnalités Flash installées avec Flash. Ces applications ont été conçues pour présenter les fonctionnalités des applications Flash aux nouveaux développeurs et illustrer le contexte de ces fonctions pour les développeurs plus expérimentés.

Pour des exemples de fichiers source axés sur le code ActionScript, reportez-vous à la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0 afin d'accéder aux exemples. Pour des fichiers exemples axés sur les composants, vous pouvez naviguer jusqu'au dossier ComponentsAS2.

Emplacement des fichiers PDF et de la documentation imprimée

Si vous préférez lire la documentation sur copie papier, les versions PDF de chaque manuel d'aide sont disponibles en téléchargement. Accédez à www.adobe.com/support/documentation_fr/ et sélectionnez le produit qui vous intéresse. Vous pouvez consulter ou télécharger le PDF ou accéder à la version LiveDocs du manuel.

Généralement, vous pouvez également acquérir la version imprimée de la documentation. Pour obtenir des informations à jour, accédez au site contenant la documentation.

Présentation de LiveDocs

Vous pouvez accéder à la documentation à partir du panneau Aide mais également sur le site Web LiveDocs. Ce site contient toutes les pages d'aide de Flash ainsi que des commentaires qui précisent, actualisent ou rectifient des parties de la documentation. Pour afficher la page équivalente sur le site Web LiveDocs, il suffit de cliquer sur Commentaires sur LiveDocs au bas d'une page du panneau Aide. Accédez à http://livedocs.adobe.com pour consulter la liste de toute la documentation disponible au format LiveDocs.

Le site Web LiveDocs est surveillé par des rédacteurs techniques. L'un des avantages de LiveDocs réside dans la possibilité de consulter des commentaires qui clarifient la documentation ou qui corrigent des problèmes rencontrés après la publication d'un logiciel. LiveDocs n'est pas un endroit où soumettre des demandes d'assistance. Par exemple, vous ne pouvez pas poser de questions au sujet de code que vous ne parvenez pas à faire fonctionner, ni commenter des problèmes de logiciel ou d'installation, ni demander comment créer quelque chose avec Flash. LiveDocs est un endroit destiné à fournir des réactions au sujet de la documentation (par exemple si vous remarquez une phrase ou un paragraphe qui mériterait d'être clarifié).

Lorsque vous cliquez sur le bouton permettant d'ajouter un commentaire dans LiveDocs, vous verrez plusieurs remarques relatives au type de commentaires admis sur le système. Lisez attentivement ces instructions, faute de quoi vous risquez de voir votre commentaire supprimé du site Web.

Si vous voulez poser une question au sujet de Flash, posez-la dans les forums Web d'Adobe : www.adobe.com/fr/support/forums/. Les forums Web sont le meilleur endroit où poser vos questions, car ils sont consultés par de nombreux employés d'Adobe, des volontaires de l'équipe Adobe, des gestionnaires et membres de groupes d'utilisateurs Adobe, voire des rédacteurs techniques.

Les ingénieurs ne contrôlent pas le système LiveDocs mais ils surveillent la liste de souhaits pour Flash. Si vous pensez avoir découvert un bogue dans votre logiciel ou si vous souhaitez demander une amélioration de Flash, veuillez remplir le formulaire approprié à l'adresse suivante : www.adobe.com/go/wish_fr. Si vous soumettez votre découverte de bogue ou votre demande d'amélioration sur LiveDocs, elle ne sera pas officiellement ajoutée à la base de données des bogues. Si vous voulez qu'un ingénieur tienne compte de votre bogue ou de votre demande, vous devez utiliser la liste de souhaits.

N'oubliez pas de faire attention aux caractères spéciaux et aux sauts de ligne lorsque vous collez du texte à partir d'Internet, y compris à partir de LiveDocs. Adobe a tout mis en oeuvre pour supprimer tous les caractères spéciaux des exemples de code mais si vous avez encore des difficultés lors du collage du code, consultez la section « Copie et collage de code », à la page 14.

Ressources en ligne supplémentaires

De nombreuses ressources en ligne offrent une abondance d'instructions, d'aides et de procédures qui vous guideront dans votre découverte de Flash. Consultez régulièrement les sites Web suivants pour être informé(e) des mises à jour :

Le Pôle de développement Adobe (www.adobe.com/fr/devnet) est régulièrement actualisé et propose les informations les plus récentes sur Flash, ainsi que des conseils d'utilisateurs expérimentés, des rubriques avancées, des exemples, des astuces, des didacticiels en plusieurs parties et d'autres mises à jour. Consultez régulièrement ce site Web pour vous tenir au courant des nouveautés de Flash et tirer le meilleur parti de votre logiciel.

Le Centre de support Adobe Flash (www.adobe.com/fr/support/flash) fournit des TechNotes, des mises à jour de la documentation et des liens vers des ressources supplémentaires dans la communauté Flash.

Le site Web des blogs Adobe (http://weblogs.adobe.com) fournit la liste des blogs des employés et de la communauté Adobe (également appelés *Weblogs*).

Les forums Adobe (http://www.adobe.com/fr/support/forums/) offrent de nombreuses discussions sur des questions spécifiques relatives à Flash, à vos applications ou au langage ActionScript. Ces forums sont surveillés par des volontaires de l'équipe Adobe et également régulièrement consultés par des employés d'Adobe. Si vous ne savez pas à qui vous adresser ou comment résoudre un problème, commencez par consulter un forum Flash.

La Communauté Adobe (http://www.adobe.com/fr/devnet/?tab:community=1) héberge régulièrement des MacroChats, séries de présentations en direct sur divers sujets et menées par les employés ou les membres de la communauté Adobe. Consultez régulièrement le site Web pour être informé(e) des mises à jour.

Documentation se rapportant à d'autres sujets

Les manuels suivants fournissent des informations supplémentaires sur des sujets généralement associés à ActionScript 2.0 :

- Pour plus d'informations sur les éléments qui composent le langage ActionScript, consultez le *Guide de référence du langage ActionScript 2.0*.
- Pour plus d'informations sur l'utilisation de l'environnement de programmation de Flash, consultez la section *Comment utiliser l'Aide*.
- Pour en savoir plus sur l'utilisation des composants, consultez la section *Utilisation des composants ActionScript 2.0*.

CHAPITRE 1

1 Nouveautés du langage ActionScript de Flash

Adobe Flash CS3 Professional bénéficie de plusieurs améliorations qui vous permettront d'écrire aisément des scripts fiables à l'aide du langage ActionScript. Ces améliorations, décrites dans le présent chapitre, portent notamment sur de nouveaux éléments de langage (consultez la section « Nouveautés dans le langage ActionScript de Flash Player 8 », à la page 23), des outils d'édition améliorés (consultez la section « Modifications de l'éditeur ActionScript », à la page 28), des changements dans les modèles de sécurité et sur d'autres améliorations apportées à cet outil de programmation dans ActionScript.

Pour plus d'informations, consultez les sections suivantes :

Nouveau dans ActionScript 2.0 et Flash Player 9.x

Flash Player 9.x permet à un utilisateur d'activer et de désactiver le mode plein écran de Flash Player. Les éléments suivants prennent en charge cette nouvelle fonctionnalité :

- Propriété `Stage.displayState`
- Gestionnaire d'événements `Stage.onFullScreen`
- Paramètre `allowFullScreen` contenu dans les balises HTML `object` et `embed`

Pour plus d'informations, consultez le *Guide de référence du langage ActionScript 2.0*.

Flash Player 9.x introduit également un nouvel événement FileReference, `onUploadCompleteData`. Cet événement peut être appelé après réception d'un chargement réussi par le serveur. Pour plus d'informations sur cet élément, consultez le *Guide de référence du langage ActionScript 2.0*.

Nouveau dans ActionScript 2.0 et Flash Player 8

Depuis son introduction il y a quelques années de cela, le langage ActionScript n'a cessé de se développer. A chaque nouvelle version de Flash, de nouveaux mots-clés, objets, méthodes et autres éléments de langage ont été ajoutés à ActionScript. Des améliorations s'articulant autour d'ActionScript sont également apportées aux environnements de programmation de Flash 8. Flash Basic 8 et Flash Professional 8 présentent plusieurs nouveaux éléments de langage dédiés aux fonctionnalités fiables, tels que les filtres et les modes de fondu, et le développement d'applications, comme par exemple l'intégration JavaScript (ExternalInterface) et le fichier d'entrée/sortie (FileReference et FileReferenceList).

Cette section présente une vue d'ensemble des éléments de langage et des classes d'ActionScript qui sont nouveaux ou modifiés dans Flash 8, ainsi que les améliorations de l'outil de programmation qui s'articulent autour d'ActionScript. Pour obtenir une liste de suppléments spécifiques dans ActionScript 2.0, consultez la section « Nouveautés dans le langage ActionScript de Flash Player 8 », à la page 23. Pour pouvoir utiliser ces nouveautés dans vos scripts, vous devez cibler Flash Player 8 lors de la publication des documents.

Flash Basic 8 et Flash Professional 8 comprennent les fonctions nouvelles suivantes (sauf avis contraire) :

- Les améliorations de l'éditeur ActionScript vous permettent de faire apparaître des éléments cachés dans vos scripts. Pour plus d'informations, consultez la section « Affichage des caractères masqués » dans le guide *Utilisation de Flash*.
- Les options de débogage sont désormais disponibles dans la fenêtre Script, ainsi que le panneau Actions, pour les fichiers ActionScript.
- Le répertoire de configuration qui comprend des fichiers XML et des fichiers de classe a été réorganisé. Voir la section « Dossiers de configuration installés avec Flash » dans le guide *Utilisation de Flash* pour plus de détails.
- Vous pouvez régler une préférence pour recharger les fichiers de script modifiés lorsque vous travaillez sur une application, ce qui vous évite de travailler avec des versions de fichiers script plus anciennes et d'écraser des fichiers script plus récents. Pour plus d'informations, consultez la section « Définition des préférences d'ActionScript » dans le guide *Utilisation de Flash*.
- La fonctionnalité de la fenêtre Script est disponible dans Flash. Ainsi, vous pouvez maintenant créer un fichier ActionScript dans l'un ou l'autre programme.

- L'Assistant de Script (similaire au mode normal dans des éditions plus anciennes de Flash) vous aide à coder sans besoin de comprendre la syntaxe. Pour plus d'informations, consultez la section « Présentation du mode Assistant de script » dans le guide *Utilisation de Flash*.
- En cours d'exécution, vous pouvez charger de nouveaux types de fichiers image, y compris des images JPEG progressives et des fichiers GIF et PNG non-animés. Si vous chargez un fichier animé, la première image de l'animation apparaît.
- Vous pouvez assigner des identifiants de liaison à des fichiers bitmap et des fichiers audio stockés dans la Bibliothèque, ainsi vous pouvez associer des images à la scène ou travailler avec ces éléments dans des bibliothèques partagées.
- La mémoire cache bitmap vous permet d'améliorer la performance de vos applications en cours d'exécution en mettant sous mémoire cache une représentation bitmap de vos occurrences. Vous pouvez utiliser le code ActionScript pour accéder à cette propriété. Pour plus d'informations, voir « A propos de la mise en cache bitmap, du défilement et des performances », à la page 474.
- La mise à l'échelle à 9 niveaux vous permet de redimmensionner les occurrences de clips sans élargir les traits du contour du clip. Vous pouvez utiliser le code ActionScript pour accéder à cette fonctionnalité dans Flash. Pour plus d'informations, voir « Utilisation de la mise à l'échelle à 9 découpes dans ActionScript », à la page 554. Pour plus d'informations sur l'accès à la mise à l'échelle à 9 niveaux dans l'outil de programmation, consultez le guide *Utilisation de Flash*.
- Vous pouvez maintenant ajouter des informations sur les métadonnées à vos fichiers FLA dans la boîte de dialogue Propriétés du document. Vous pouvez ajouter un nom et une description à votre fichier FLA en utilisant la boîte de dialogue pour une meilleure visibilité de recherche en ligne.
- Le panneau Chaînes a été amélioré et inclut désormais une prise en charge multiligne dans le champ Chaîne et dans le fichier de langue XML. Pour plus d'informations, voir « panneau », à la page 445.
- Un nouveau nettoyeur de mémoire a été intégré dans Flash Player, qui utilise un nettoyeur incrémentiel pour améliorer les performances.
- Le rendement de création d'applications accessibles a été amélioré. Avec Flash Player 8, les développeurs n'ont plus besoin d'ajouter tous les objets à l'index de tabulation pour que le contenu soit lu correctement avec un lecteur d'écran. Pour plus d'informations sur les index de tabulation, consultez les entrées `tabIndex (propriété Button.tabIndex)`, `tabIndex (propriété MovieClip.tabIndex)` et `tabIndex (propriété TextField.tabIndex)` dans le *Guide de référence du langage ActionScript 2.0*.

- Flash Player a amélioré la sécurité des fichiers locaux pour plus de sécurité lors de l'exécution de fichiers SWF sur votre disque dur. Pour plus d'informations sur la sécurité des fichiers locaux, consultez « Sécurité des fichiers locaux et Flash Player », à la page 677.
- En utilisant le code ActionScript, aidez-vous de l'API de dessin pour contrôler le style de traits que vous dessinez. Pour plus d'informations sur les nouveaux styles de lignes, consultez la section « Utilisation de styles de ligne », à la page 543.
- En utilisant le code ActionScript, aidez-vous de l'API de dessin pour créer des dégradés plus complexes avec lesquels vous remplissez des formes. Pour plus d'informations sur les remplissages de dégradés, consultez la section « Utilisation de remplissages dégradés complexes », à la page 542.
- Utilisez le code ActionScript pour appliquer plusieurs filtres à des objets sur la scène (tels que des occurrences de clips). Pour plus d'informations sur les filtres et sur ActionScript, consultez la section « Utilisation des filtres avec ActionScript », à la page 499.
- Utilisez FileReference et FileReferenceList API pour télécharger des fichiers sur un serveur. Pour plus d'informations, voir « Téléchargement de fichier depuis et vers le serveur », à la page 643.
- Utilisez le code ActionScript pour accéder à de nouveaux moyens avancés d'application et de manipulation des couleurs. Pour plus d'informations, consultez les sections « Définition des valeurs de couleurs », à la page 568 et `ColorTransform (flash.geom.ColorTransform)` dans le *Guide de référence du langage ActionScript 2.0*.
- De nombreuses améliorations ont été apportées au traitement de texte, y compris de nouvelles options, propriétés et paramètres dans les classes de champ de texte et de format de texte. Pour plus d'informations, consultez les sections `TextField` et `TextFormat` du *Guide de référence du langage ActionScript 2.0*.
- Utilisez le code ActionScript pour accéder aux fonctionnalités avancées d'anti-aliasing. Pour plus d'informations, voir « Présentation du rendu d'un texte anti-alias », à la page 395.
- Vous pouvez supprimer les fichiers ASO quand vous testez votre application. Sélectionnez Contrôle > Supprimer les fichiers ASO ou Contrôle > Supprimer les fichiers ASO et tester le clip dans l'outil de programmation. Pour plus d'informations, consultez la section « Utilisation des fichiers ASO », à la page 259.

Pour obtenir une liste de classes, d'éléments de langage, de méthodes et de propriétés spécifiques ajoutées à ActionScript 2.0 dans Flash 8, consultez la section « Définition des préférences d'ActionScript » dans le guide *Utilisation de Flash*.

Nouveautés dans le langage ActionScript de Flash Player 8

Les classes et les éléments de langage qui suivent sont des nouveautés ou prises en charge dans Flash Player 8.

Les classes suivantes sont des nouveautés d'ActionScript 2.0 dans Flash 8 :

- La classe BevelFilter (du paquet flash.filters) vous permet d'ajouter des effets de biseau aux objets.
- La classe BitmapData (du paquet flash.display) vous permet de créer des images bitmap transparentes ou opaques dimensionnées de manière arbitraire et de les manipuler à votre guise lors de l'exécution.
- La classe BitmapFilter (du paquet flash.display) est une classe de base pour les effets de filtres.
- La classe BlurFilter vous permet d'appliquer un effet visuel de flou aux objets dans Flash.
- La classe ColorMatrixFilter (du paquet flash.filters) vous permet de transformer les couleurs ARGB et les valeurs alpha.
- La classe ColorTransform (du paquet flash.geom) vous permet de régler les valeurs des couleurs dans un clip. La classe Color est déconseillée et doit être remplacée par la classe ColorTransform.
- La classe ConvolutionFilter (du paquet flash.filters) vous permet d'appliquer des effets de filtre de convolution matricielle.
- La classe DisplacementMapFilter (du paquet flash.filters) vous permet d'utiliser des valeurs de pixel depuis un objet BitmapData pour déplacer un objet.
- La classe DropShadowFilter (du paquet flash.filters) vous permet d'ajouter des projections d'ombres à des objets.
- La classe ExternalInterface (du paquet flash.external) vous permet de communiquer en utilisant ActionScript avec le conteneur Flash Player (le système supportant l'application Flash, par exemple une page HTML utilisant JavaScript ou une application de bureau).
- La classe FileReference (du paquet flash.net) vous permet de charger et télécharger des fichiers entre l'ordinateur d'un utilisateur et le serveur.
- La classe FileReference (du paquet flash.net) vous permet de sélectionner un ou plusieurs fichiers à charger.
- La classe GlowFilter (du paquet flash.filters) vous permet d'ajouter des effets de brillance à des objets.
- La classe GradientBevelFilter (du paquet flash.filters) vous permet d'ajouter des effets de biseaux dégradés à des objets.

- La classe GradientGlowFilter (du paquet flash.filters) vous permet d'ajouter des effets de brillance dégradée à des objets.
- La classe IME (de la classe System) vous permet de manipuler directement l'IME (Input Method Editor) du système d'exploitation sous lequel l'application Flash Player s'exécute sur l'ordinateur client.
- La classe Locale (du paquet mx.lang) vous permet de contrôler la façon dont le texte multilingue s'affiche dans un fichier SWF.
- La classe Matrix (du paquet flash.geom) représente une matrice de transformation qui détermine la façon de mapper des points d'un espace de coordonnées à l'autre.
- La classe Point (du paquet flash.geom) représente un emplacement dans un système de coordonnées à deux dimensions (*x* est l'axe horizontal et *y* représente l'axe vertical).
- La classe Rectangle (du paquet flash.geom) vous permet de créer et de modifier des objets rectangles.
- La classe TextRenderer (du paquet flash.text) permet d'exploiter la fonction d'anti-aliasing avancé des polices incorporées.
- La classe Transform (du paquet flash.geom) rassemble des données sur les transformations de couleurs et coordonne les manipulations que vous appliquez à un objet MovieClip.

REMARQUE	Une prise en charge officielle a été ajouté à la classe AsBroadcaster dans Flash 8.

Les nouveautés dans les éléments de langage, les méthodes et les fonctions dans les classes existantes d'ActionScript comprennent notamment :

- La fonction globale `showRedrawRegions` permet au débogueur de détourer les zones de l'écran qui sont redessinées (c'est-à-dire des zones obsolètes qui sont mises à jour). La fonction permet à l'utilisateur de montrer ce qui a été redessiné, mais ne vous permet pas de redessiner les zones.
- La propriété `blendMode` de la classe Button, qui règle le mode de fondu pour l'occurrence de bouton.
- La propriété `cacheAsBitmap` de la classe Button, qui vous permet de cacher l'objet en tant que représentation bitmap interne de l'occurrence.
- La propriété `filters` de la classe Button, qui est un tableau indexé contenant chacun des objets filtre associés au bouton.
- La propriété `scale9Grid` de la classe Button, qui est la zone rectangulaire définissant les neuf zones de redimensionnement du bouton.

- La propriété `hasIME` de la classe System.capabilities, qui indique si le système possède un IME.
- La propriété `getUTCYear` de la classe Date, qui renvoie l'année de l'objet Date spécifié, conformément à l'heure locale.
- La méthode `isAccessible()` de la classe Key retourne une valeur booléenne qui indique si d'autres fichiers SWF peuvent accéder à la dernière touche enfoncée, en fonction des restrictions de sécurité.
- Le programme de traitement d'événement `onHTTPStatus` de la classe LoadVars retourne le code d'état qui est retourné par le serveur (par exemple, la valeur 404 pour impossible de trouver la page). Pour plus d'informations, consultez `HTTPStatus (gestionnaire LoadVars.onHTTPStatus)` dans le *Guide de référence du langage ActionScript 2.0*.
- La méthode `attachBitmap()` de la classe MovieClip, qui associe une image bitmap à un clip. Pour plus d'informations, consultez la section `BitmapData (flash.display.BitmapData)` dans le *Guide de référence du langage ActionScript 2.0*.
- La méthode `beginBitmapFill()` de la classe MovieClip, qui remplit un clip avec une image bitmap.
- Les paramètres de `spreadMethod`, `interpolationMethod` et de `focalPointRatio` de la méthode `beginGradientFill()`de la classe MovieClip. Cette méthode remplit une zone de dessin avec une image bitmap et le bitmap peut être répété ou former une mosaïque afin de remplir la zone.
- La propriété `blendMode` de la classe MovieClip, qui règle le mode de fondu pour l'occurrence.
- La propriété `cacheAsBitmap` de la classe MovieClip, qui vous permet de cacher l'objet en tant qu'une représentation bitmap interne de l'occurrence.
- La propriété `filters` dans la classe MovieClip, qui est un tableau indexé contenant chacun des objets filtre associés à l'occurrence.
- La méthode `getRect()` de la classe MovieClip, qui renvoie les propriétés aux valeurs de coordonnées minimales et maximales de l'occurrence donnée.
- La méthode `lineGradientStyle()` de la classe MovieClip, qui spécifie un style de ligne dégradée que Flash utilise lors du dessin d'un chemin.
- Les paramètres `pixelHinting`, `noScale`, `capsStyle`, `jointStyle` et `miterLimit` de la méthode `lineStyle()`de la classe MovieClip. Ces paramètres spécifient les types de styles de lignes que vous pouvez utiliser en dessinant des lignes.
- La propriété `opaqueBackground` de la classe MovieClip, qui règle la couleur de l'arrière-plan opaque (non transparent) du clip à la couleur spécifiée par une valeur RVB hexadécimale.

- La propriété `scale9Grid` de la classe MovieClip, qui est la zone rectangulaire définissant les neuf zones de redimensionnement de l'occurrence.
- La propriété `scrollRect` de la classe MovieClip, qui vous permet de parcourir rapidement le contenu d'un clip et d'ouvrir une fenêtre capable d'afficher davantage de contenu.
- La propriété `transformation` de la classe MovieClip, qui vous permet des réglages se rapportant à la matrice d'un clip, à la transformation des couleurs et aux limites des pixels. Pour plus d'informations, consultez `Transform (flash.geom.Transform)` dans le *Guide de référence du langage ActionScript 2.0*.
- Le paramètre `statut` de `MovieClipLoader`. Le programme de traitement d'événements `onLoadComplete` renvoie le code de statut qui est ramené depuis le serveur (par exemple, la valeur 404 pour impossible de trouver la page). Pour plus d'informations, consultez `onLoadComplete (écouteur d'événement MovieClipLoader.onComplete)` dans le *Guide de référence du langage ActionScript 2.0*.
- Le programme de traitement d'événements `onLoadError` de la classe MovieClipLoader est appelé lorsque le chargement d'un fichier chargé avec `MovieClipLoader.loadClip()` a échoué.
- Le paramètre `secure` de la méthode `SharedObject.getLocal()` détermine si l'accès à cet objet partagé est restreint aux fichiers SWF acheminés à travers une connexion HTTPS. Pour plus d'informations, consultez `getLocal (méthode SharedObject.getlocal)` dans le *Guide de référence du langage ActionScript 2.0*.
- La propriété de type sandbox de la classe System.security indique le type de sandbox de sécurité dans lequel le SWF appelant fonctionne. Pour plus d'informations, consultez la section `sandboxType (propriété security.sandboxType)` du *Guide de référence du langage ActionScript 2.0*.
- La propriété `antiAliasType` de la classe TextField, qui règle le type d'anti-aliasiang avancé que vous utilisez pour l'occurrence TextField.
- La propriété `filters` de la classe TextField, qui est un tableau indexé contenant chacun des objets filtre associés à la zone de texte.
- La propriété `gridFitType` de la classe TextField, qui règle le type de grille que vous utilisez pour l'occurrence. Pour plus d'informations sur les grilles et sur TextField.gridFitType, consultez la section `gridFitType (propriété TextField.gridFitType)` dans le *Guide de référence du langage ActionScript 2.0*.
- La propriété `sharpness` de la classe TextField, qui règle le type de définition des bords de glyphes pour l'occurrence zone de texte. Vous devez régler la méthode `antiAliasType()` en mode avancé si vous utilisez cette propriété.

- La propriété `thickness` de la classe TextField, qui règle l'épaisseur des bords de glyphes pour l'occurrence zone de texte. Vous devez régler la méthode `antiAliasType()` en mode avancé si vous utilisez cette propriété.
- La valeur `justify` pour la propriété `align` de la classe TextFormat, qui vous permet de justifier un paragraphe donné.
- La propriété `indent` de la classe TextFormat, qui vous permet d'utiliser des valeurs négatives.
- La propriété `kerning` de la classe TextFormat, qui vous permet d'activer ou de désactiver le crénage pour l'objet TextFormat.
- La propriété `leading` de la classe TextFormat, qui vous permet d'utiliser un lignage négatif, afin que l'espace entre les lignes soit inférieur à la hauteur de texte. Ceci vous permet de placer des lignes de texte proches les unes des autres dans vos applications.
- La propriété `letterSpacing` de la classe TextFormat, qui vous permet de spécifier la taille de l'espace uniformément réparti entre les caractères.
- La propriété `_alpha` de la classe Video, qui est le montant spécifié de transparence pour l'objet vidéo.
- La propriété `_height` de la classe Video, qui indique la hauteur de l'occurrence vidéo.
- La propriété `_name` de la classe Video, qui indique le nom de l'occurrence vidéo.
- La propriété `_parent` de la classe Video, qui indique l'occurrence de clip ou l'objet qui contient l'occurrence vidéo.
- La propriété `_rotation` de la classe Video, qui vous permet de régler le mouvement de rotation de l'occurrence vidéo en degrés.
- La propriété `_visible` de la classe Video, qui vous permet de régler la visibilité d'une occurrence vidéo.
- La propriété `_width` de la classe Video, qui vous permet de régler la largeur de l'occurrence vidéo.
- La propriété `_x` de la classe Video, qui vous permet de régler les coordonnées *x* de l'occurrence vidéo.
- La propriété `_xmouse` de la classe Video, qui vous permet de régler les coordonnées *x* de la position du pointeur de la souris.
- La propriété `_xscale` de la classe Video, qui vous permet de régler le pourcentage de redimensionnement horizontal de l'occurrence vidéo.
- La propriété `_y` de la classe Video, qui vous permet de régler les coordonnées *y* de l'occurrence vidéo.
- La propriété `_ymouse` de la classe Video, qui vous permet de régler les coordonnées *y* de la position du pointeur de la souris.

- La propriété `_yscale` de la classe Video, qui vous permet de régler le pourcentage de redimensionnement vertical de l'occurrence vidéo.
- Le programme de traitement d'événement `onHTTPStatus` de la classe XML retourne le code de statut qui est revenu du serveur (par exemple, la valeur 404 pour impossible de trouver la page). Pour plus d'informations, consultez la section `onHTTPStatus (gestionnaire XML.onHTTPStatus)` du *Guide de référence du langage ActionScript 2.0*.
- La propriété `localName` de la classe XMLNode, qui renvoie le nom complet de l'objet noeud XML (y compris le préfixe et le nom local).
- La propriété `namespaceURI`de la classe XMLNode, qui lit l'URI de l'espace de nom résultant du préfixe du nœud XML. Pour plus d'informations, consultez la section `namespaceURI (propriété XMLNode.namespaceURI)` du *Guide de référence du langage ActionScript 2.0*.
- La propriété `prefix` de la classe XMLNode, qui lit le préfixe du nom du nœud.
- La méthode `getNamespaceForPrefix()` de la classe XMLNode, qui renvoie le nom d'espace URI se rapportant au préfixe spécifié pour le nœud.
- La méthode `getPrefixForNamespace` de la classe XMLNode, qui renvoie le préfixe se rapportant à un nom d'espace URI donné pour le nœud.

À propos des éléments de langage déconseillés

Certains éléments de langage sont déconseillés dans Flash Player 8. Pour obtenir une liste des éléments de langage déconseillés et des méthodes de remplacement à utiliser dans Flash Player 8, consultez les sections suivantes dans le *Guide de référence du langage ActionScript 2.0* :

- `Classes déconseillées`
- `Fonctions déconseillées`
- `Propriétés déconseillées`
- `Opérateurs déconseillés`

Modifications de l'éditeur ActionScript

L'éditeur ActionScript dans le panneau Actions et dans la fenêtre Script a subi différentes mises à jour afin d'être plus fiable et plus simple à utiliser que dans les versions antérieures de l'outil. Cette section récapitule toutes ces modifications.

Affichage des caractères masqués Pour activer ou désactiver l'affichage des caractères masqués, vous pouvez désormais utiliser le menu d'options contextuel de la fenêtre de script, du panneau Débogueur et du panneau de sortie. Pour plus d'informations sur cette fonctionnalité, consultez le guide *Utilisation de Flash*.

L'Assistant de script a été ajouté au panneau Actions Dans les versions antérieures de Flash, vous pouviez travailler dans le panneau Actions soit en *mode normal* dans lequel vous complétiez des options et des paramètres pour créer du code, soit en *mode expert* dans lequel vous ajoutiez des commandes directement dans la fenêtre de script. Ces options n'étaient plus disponibles dans Flash MX 2004 ou Flash MX Professional 2004. Flash Basic 8 et Flash Professional 8 permettent cependant d'utiliser *le mode d'Assistant de script*, semblable (mais plus fiable) au mode normal. Pour plus d'informations et obtenir un didacticiel sur l'Assistant Script, consultez le guide *Utilisation de Flash*.

Rechargement des fichiers modifiés Vous pouvez recharger les fichiers script modifiés lorsque vous travaillez sur une application. Un message d'avertissement apparaît pour vous inviter à recharger les fichiers script modifiés qui sont liés à l'application sur laquelle vous travaillez. Cette fonctionnalité est particulièrement profitable aux équipes qui travaillent sur des applications en même temps, car elle vous évite de travailler avec des scripts périmés ou d'écraser des versions plus récentes d'un script. Si un fichier de script a été déplacé ou supprimé, un message d'avertissement apparaît et vous rappelle d'enregistrer les fichiers selon les besoins. Pour plus d'informations, consultez la section « Définition des préférences d'ActionScript » dans le guide *Utilisation de Flash*.

Modifications du modèle de sécurité pour les fichiers SWF installés localement

Flash Player 9.x prend également en charge une nouvelle balise HTML, `allowNetworking`. Pour plus d'informations, consultez le chapitre Sécurité du manuel *Programmation avec ActionScript 3.0*.

Flash Player 8 a introduit un nouveau modèle de sécurité améliorée dans lequel les applications Flash et les fichiers SWF sur un ordinateur local peuvent communiquer par Internet sur le système de fichiers local plutôt que de passer par un serveur Internet à distance. Lors de la programmation d'une application Flash, il faut indiquer si un fichier SWF est autorisé à communiquer avec un réseau ou un système de fichiers local.

REMARQUE	Dans le cadre de cette section, un *fichier SWF local* est un fichier SWF installé localement sur l'ordinateur d'un utilisateur, et non pas obtenu via un site Web, et qui ne comprend pas de fichiers de projections (EXE).

Dans les versions antérieures de Flash Player, les fichiers SWF locaux pouvaient interagir avec d'autres fichiers SWF et charger des données depuis un ordinateur à distance ou local sans effectuer de réglages de sécurité. Dans Flash Player 8 et les versions ultérieures, un fichier SWF ne peut pas établir de connexion entre le système de fichiers local et le réseau (tel qu'Internet) dans une même application sans effectuer de réglage de sécurité. Ceci est pour votre sécurité ; c'est ainsi qu'un fichier SWF ne peut pas lire de fichiers sur votre disque dur, puis envoyer le contenu de ces fichiers sur Internet.

Cette restriction de sécurité affecte un contenu déployé localement, qu'il soit patrimonial (un fichier FLA créé dans une version antérieure de Flash) ou qu'il ait été créé dans Flash 8 et ses versions ultérieures. En utilisant l'outil Flash MX 2004 ou une version antérieure, vous pouviez tester une application Flash qui s'exécutait localement et accédait également à Internet. Dans Flash Player 8 et les versions ultérieures, cette application demande désormais à l'utilisateur l'autorisation de communiquer avec Internet.

Quand vous testez un fichier sur votre disque dur, plusieurs étapes permettent de déterminer si le fichier est un document local fiable (sûr) ou potentiellement incertain (dangereux). Si vous créez le fichier dans l'environnement autorisé Flash (par exemple, quand vous sélectionnez Contrôle > Tester le clip), votre fichier est considéré comme fiable parce qu'il se trouve dans l'environnement de test.

Dans la version Flash Player 7 et les versions antérieures, les fichiers SWF locaux étaient autorisés à accéder au réseau et au système de fichiers local. Dans Flash Player 8 et les versions ultérieures, les fichiers SWF locaux peuvent disposer de trois niveaux d'autorisation :

- Accès au système de fichiers local uniquement (niveau par défaut). Le fichier SWF local peut lire le système de fichiers local et les chemins de réseau de la convention UNC, mais il ne peut pas communiquer avec Internet.
- Accès au réseau uniquement. Le fichier SWF local ne peut accéder qu'au réseau (tel qu'Internet) et pas au système de fichiers local sur lequel il est installé.
- Accès au système de fichiers local et au réseau. Le fichier SWF local peut lire le système de fichiers local sur lequel il est installé, lire et écrire sur les serveurs qui lui accordent une autorisation et peut inter-coder d'autres fichiers SWF du réseau ou du système de fichiers local qui l'y autorisent.

Pour plus de détails sur chaque niveau d'autorisation, consultez la section « Sécurité des fichiers locaux et Flash Player », à la page 677.

Des retouches légères ont également été apportées à System.security.allowDomain et des améliorations à System.security.allowInsecureDomain. Pour plus d'informations sur la sécurité des fichiers locaux, consultez le Chapitre 16, « Fonctionnement de la sécurité »

CHAPITRE 2

A propos d'ActionScript

2

Les fonctionnalités de programmation orientée objet (OOP) d'ActionScript 2.0 reposent sur la norme ECMAScript 4 Draft Proposal en cours de développement par ECMA TC39-TG1 (voir le site www.mozilla.org/js/language/es4/index.html). Dans la mesure où la proposition de la ECMA-4 n'est pas encore une norme établie car elle est encore en cours d'élaboration, ActionScript 2.0 ne l'applique pas de façon stricte.

ActionScript 2.0 prend en charge tous les éléments standard du langage ActionScript. Il vous permet simplement d'écrire des scripts plus proches des normes utilisées par d'autres langages orientés objet, tels que Java. ActionScript 2.0 intéressera principalement les développeurs Flash de niveau intermédiaire ou expert qui créent des applications nécessitant l'implémentation de classes et de sous-classes. ActionScript 2.0 vous permet également de déclarer le type d'objet d'une variable au moment de sa création (consultez « Affectation des types de données et typage strict », à la page 45) et améliore considérablement les erreurs de compilateur (consultez l'Annexe A, « Messages d'erreur », à la page 759).

Les points principaux à connaître sur ActionScript 2.0 sont les suivants :

- Les scripts utilisant ActionScript 2.0 pour définir des classes ou des interfaces doivent être enregistrés en tant que fichiers de script externes, avec une seule classe définie par script. Les classes et les interfaces ne peuvent donc pas être définies dans le panneau Actions.
- Vous pouvez importer des fichiers de classe individuels de façon implicite (en les enregistrant dans un emplacement spécifié par des chemins de recherche généraux ou de niveau document et en les utilisant dans un script) ou de façon explicite (en utilisant la commande `import`). Vous pouvez importer des lots de fichiers (collections de fichiers de classe placés dans un répertoire) en utilisant des caractères génériques.
- Les applications développées avec ActionScript 2.0 sont prises en charge par Flash Player 6 et les versions ultérieures.

ATTENTION

Le paramètre de publication par défaut des nouveaux fichiers créés dans Flash 9 est ActionScript 3.0. Si vous envisagez de modifier un fichier FLA existant à l'aide d'ActionScript 1.0 ou ActionScript 2.0 pour utiliser la syntaxe ActionScript 2.0, vérifiez que le fichier FLA indique bien ActionScript 2.0 dans ses paramètres de publication. Si ce n'est pas le cas, votre fichier ne sera pas compilé correctement, même si Flash ne génère pas d'erreurs de compilation.

Pour plus d'informations sur l'utilisation d'ActionScript 2.0 dans la rédaction de programmes orientés objet dans Flash, consultez le Chapitre 6, « Classes », à la page 197.

Bien qu'Adobe recommande d'utiliser ActionScript 3.0, vous pouvez continueer à utiliser la syntaxe ActionScript 1.0 et ActionScript 2.0.

Présentation d'ActionScript

Les fonctions principales d'ActionScript 2.0 sont regroupées ci-dessous :

Modèle familier de programmation orientée objet (OOP) ActionScript 2.0 offre un modèle connu de tous pour la création de programmes orientés objet. ActionScript 2.0 met en oeuvre plusieurs concepts et mots-clés orientés objet, tels que *class, interface* et *packages* qui vous sembleront familiers si vous avez déjà programmé en Java.

Le modèle OOP d'ActionScript 2.0 est une « formalisation syntaxique » de la méthode de chaînage de prototype utilisée dans les précédentes versions de Flash pour créer des objets et établir une relation d'héritage. Avec ActionScript 2.0, vous pouvez créer des classes personnalisées et étendre les classes intégrées de Flash.

Typage strict des données ActionScript 2.0 vous permet également de spécifier explicitement les types de données pour les variables et pour les paramètres et les renvois des fonctions. Par exemple, le code suivant déclare une variable appelée `userName` de type chaîne (type de données ActionScript intégré ou classe).

```
var userName:String = "";
```

Avertissements et messages d'erreur du compilateur Les deux précédentes fonctions (modèle OOP et typage strict des données) permettent à l'outil de programmation et au compilateur de générer des avertissements et messages d'erreur qui vous aident à localiser les bogues de vos applications dans Flash plus rapidement qu'auparavant.

Lorsque vous utilisez ActionScript 2.0, assurez-vous que les paramètres de publication du fichier FLA spécifient ActionScript 2.0 (le paramètre par défaut pour Flash Player 9 est ActioScript 3.0). En outre, si vous ouvrez un ancien fichier FLA qui utilise ActionScript 1.0 et que vous le modifiez en langage ActionScript 2.0, vous devez régler les paramètres de publication du fichier FLA sur ActionScript 2.0. Sinon, votre fichier FLA ne sera pas compilé correctement et aucune erreur ne sera générée.

Différences entre ActionScript 1.0 et ActionScript 2.0

Lorsque vous démarrez avec un document ou une application dans Flash, vous devez déterminer le mode d'organisation des fichiers qui lui sont associés. Vous pouvez utiliser des classes dans certains projets, comme lorsque vous créez des applications ou des fichiers FLA complexes, bien que certains documents excluent l'utilisation de classes. Ainsi, de nombreux exemples courts de cette documentation n'ont pas recours à des classes. L'utilisation de classes pour stocker des fonctionnalités n'est pas la meilleure solution ni la plus facile pour les petites applications ou les fichiers FLA simples. Il est généralement plus efficace de placer du code ActionScript dans le document. Essayez alors de placer l'ensemble du code sur le scénario et sur le moins d'images possible. Evitez de placer du code sur ou dans les occurrences (telles que les boutons ou les clips) dans un fichier FLA.

Lorsque vous créez un petit projet, il est souvent plus difficile d'utiliser des classes ou des fichiers de code externes pour organiser le code ActionScript que d'ajouter ce code au sein du fichier FLA. Il est parfois plus facile de regrouper l'ensemble du code ActionScript au sein du fichier FLA que de le placer dans une classe à importer. Cela ne signifie pas que vous devez nécessairement utiliser ActionScript 1.0. Vous pouvez décider de placer votre code dans le fichier FLA à l'aide d'ActionScript 2.0 avec son typage strict des données et ses nouvelles méthodes et propriétés. ActionScript 2.0 propose également une syntaxe qui applique les normes des autres langages de programmation. Le langage devient alors plus aisé et plus utile à apprendre. Par exemple, il vous sera plus facile de vous familiariser avec ActionScript si vous connaissez déjà un autre langage reposant sur les mêmes normes de structure et de syntaxe. De même, vous pourrez appliquer les connaissances que vous allez acquérir à d'autres langages de programmation. ActionScript 2.0 vous permet de bénéficier d'une approche orientée objet pour développer des applications en utilisant un ensemble supplémentaire d'éléments de langage, ce qui peut être avantageux pour le développement de votre application.

Dans certains cas, il n'est pas possible de choisir la version d'ActionScript à utiliser. Si vous créez un fichier SWF destiné à une ancienne version de Flash Player, telle qu'une application pour périphérique mobile, choisissez ActionScript1.0 pour sa compatibilité.

Néanmoins, quelle que soit la version d'ActionScript, vous devez appliquer les meilleures pratiques. La plupart de ces pratiques, telles qu'un usage cohérent des majuscules, la saisie automatique, l'amélioration de la lisibilité, l'exclusion de mots clés pour les noms d'occurrence et l'application d'une convention d'appellation cohérente s'appliquent aux deux versions.

Si vous prévoyez de mettre à jour votre application avec les versions suivantes de Flash ou si vous devez la développer et créer du code plus complexe, utilisez ActionScript 2.0 et les classes afin de faciliter la mise à jour et la modification de vos applications.

Présentation d'ActionScript et de Flash Player

Si vous compilez un fichier SWF contenant du code ActionScript 2.0 dont l'option Paramètres de publication est définie sur Flash Player 6 et ActionScript 1.0, votre code reste valable tant qu'il n'utilise pas les classes ActionScript 2.0. Le code n'est pas sensible à la casse tandis que Flash Player l'est. Cependant, si vous compilez votre fichier SWF avec l'option Paramètres de publication définie sur Flash Player 7 ou 8 et ActionScript 1.0, Flash impose le respect de la casse.

Annotations de type de données (types de données strictes) sont imposées sur un au moment de la compilation pour Flash Player 7 et versions ultérieures quand les paramètres de publication sont définis sur ActionScript 2.0.

ActionScript 2.0 compile en pseudo-code ActionScript 1.0 lorsque vous publiez vos applications, et vous pouvez ainsi cibler Flash Player 6 et versions ultérieures lorsque vous travaillez avec ActionScript 2.0.

CHAPITRE 3

Données et types de données

3

Ce chapitre est le premier d'une longue série qui décrit les concepts fondamentaux d'ActionScript. Pour apprendre à créer des applications complexes, vous vous exercerez à l'utilisation des techniques de base de la programmation. Dans ce chapitre, vous apprendrez également à manipuler les données des fichiers FLA et à identifier les données compatibles avec vos applications. Dans le chapitre suivant, Chapitre 4, « Eléments fondamentaux du langage et de la syntaxe », vous découvrirez les instructions des formulaires et la syntaxe d'ActionScript. Ensuite, le Chapitre 5, « Fonctions et méthodes » présente l'utilisation des fonctions et des méthodes dans le langage ActionScript.

Pour plus d'informations sur les données et les types de données, consultez les sections suivantes :

A propos des données

Le terme *données* fait référence aux nombres, chaînes de caractères et autres informations qui sont manipulés au sein de Flash. Lorsque vous créez des applications ou des sites Internet, la manipulation d'un certain nombre de données est inévitable. Vous utilisez également des données pour créer des graphiques sophistiqués et des animations générées par des scripts. Dans ce cas, vous devez manipuler les valeurs utilisées pour appliquer vos effets spéciaux.

Vous pouvez définir des données en *variables* dans Flash ou les charger à partir de fichiers externes ou de sites utilisant XML, des services Web, des classes ActionScript intégrées, etc. Vous pouvez stocker les données dans une base de données, puis les représenter sous différentes formes dans un fichier SWF. Cela comprend l'affichage des informations dans des champs de texte ou des composants et l'affichage des images dans des occurrences de clip.

Les types de données les plus utilisés sont les suivants : chaînes (série de caractères, telles que des noms et des phrases), nombres, objets (tels que des clips), valeurs booléennes (`true` et `false`), etc. Dans ce chapitre, vous apprendrez également à reconnaître les différents types de données dans Flash et à les utiliser.

Pour plus d'informations sur les types de données, consultez la section « Présentation des types de données », à la page 36. Pour plus d'informations sur les variables, consultez la section « Présentation des variables », à la page 51.

Présentation des types de données

Le *type de données* décrit une information et les sortes d'opération qu'elles peuvent subir. Vous stockez les données dans des variables. Vous utilisez les types de données lors de la création des variables, des occurrences d'objets et des définitions de fonctions pour attribuer le type de données que vous manipulez. Lorsque vous écrivez du code ActionScript, vous utilisez de nombreux types de données différents.

ActionScript 2.0 définit plusieurs types de données fréquemment utilisés. Les types de données décrivent le genre de valeur qu'une variable ou qu'un élément ActionScript peut contenir. Après l'affectation de son type de données, une variable ne peut contenir qu'une valeur correspondant à ce type. Pour plus d'informations sur les variables, consultez la section « Présentation des variables », à la page 51.

ActionScript dispose de nombreux types de données de base que vous utiliserez fréquemment dans vos applications. Pour plus d'informations, consultez le tableau de la section « Types de données complexes et primitifs », à la page 37.

ActionScript dispose également de classes de base, telles que Array et Date, qui sont considérées comme des types de données complexes ou de référence. Pour plus d'informations sur les types de données complexes ou de référence, consultez la section « Types de données complexes et primitifs », à la page 37. De plus, tous les types de données et les classes sont intégralement définis dans le *Guide de référence du langage ActionScript de Flash*.

Vous pouvez également créer des classes personnalisées pour vos applications. Toute classe définie par une déclaration est également considérée comme un type de données. Pour plus d'informations sur les classes intégrées ou de base, consultez la section « Présentation des classes de niveau supérieur et intégrées », à la page 263. Pour plus d'informations sur la création de classes personnalisées, consultez le Chapitre 6, « Classes », à la page 197.

Dans ActionScript 2.0, vous pouvez affecter des types de données aux variables lorsque vous les déclarez. Les types de données que vous affectez peuvent être de type de base ou peuvent représenter une classe personnalisée que vous avez créée. Pour plus d'informations, voir « Affectation des types de données et typage strict », à la page 45.

Lorsque vous déboguez des scripts, vous pouvez avoir besoin de déterminer le type de données d'une expression ou d'une variable pour comprendre son comportement. Vous pouvez effectuer cette opération avec les opérateurs `instanceof` et `typeof` (consultez la section « Identification du type de données », à la page 50).

Lors de l'exécution, vous pouvez convertir un type de données en un autre en utilisant l'une des fonctions de conversion suivantes : `Array()`, `Boolean()`, `Number()`, `Object()`, `String()`.

Pour un exemple du fichier datatypes.fla, qui vous explique comment utiliser des types de données dans une application, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/DataTypes afin d'accéder à l'exemple.

Types de données complexes et primitifs

Les valeurs des différents types de données se divisent en deux catégories principales : *primitive* et *complexe*.

Une *valeur primitive* (ou type de données primitif) est une valeur stockée par ActionScript au niveau d'abstraction le plus bas, ce qui signifie que les opérations qu'elles subissent sont généralement plus rapides et plus importantes que celles effectuées sur les types de données complexes. Les types de données suivants définissent tous un ensemble de valeurs primitives : Boolean, null, Number, String et undefined.

Une *valeur complexe* (ou type de données complexe) est une valeur non primitive qui fait référence aux valeurs primitives. Elles sont souvent appelées types de données de *référence*. Les valeurs complexes appartiennent au type de données Objet ou à un type de données basé sur le type de données Objet. Les types de données qui définissent des ensembles de valeurs complexes comprennent Array, Date, Error, Function et XML. Pour plus d'informations sur ces types de données complexes, consultez les entrées correspondantes dans le *Guide de référence du langage ActionScript 2.0*.

Les variables contenant des données primitives ont un comportement différent de celles contenant des données complexes dans certaines situations. Pour plus d'informations, voir « Utilisation des variables dans un projet », à la page 72.

ActionScript dispose des types de données de base suivants que vous pouvez utiliser fréquemment dans vos applications :

Type de données	Description
Boolean	Primitive. Le type booléen est composé de deux valeurs : `true` et `false`. Les variables de ce type ne prennent en charge aucune autre valeur. La valeur par défaut d'une variable booléenne qui a été déclarée mais non initialisée est `false`. Pour plus d'informations, voir « Type de données Boolean », à la page 39.
MovieClip	Complexe. Le type de données MovieClip vous permet de contrôler les symboles de clip au moyen des méthodes de la classe MovieClip. Pour plus d'informations, voir « Type de données MovieClip », à la page 40.
null	Primitive. Ce type de données ne contient que la valeur `null`. Cette valeur signifie « pas de valeur », c'est-à-dire une absence de données. Vous pouvez affecter la valeur `null` dans de nombreuses situations pour indiquer qu'une propriété ou une variable n'a pas reçu de valeur. Il s'agit du type de données par défaut pour toutes les classes qui définissent des types de données complexes. La classe Objet est la seule exception à cette règle, car elle est `undefined` par défaut. Pour plus d'informations, voir « Type de données Null », à la page 42.
Number	Primitive. Ce type de données peut représenter des entiers, des entiers non signés et des nombres en virgule flottante. Pour stocker un nombre en virgule flottante, vous devez y insérer une virgule. Sans virgule, le nombre est considéré comme un entier. Le type de données Numérique peut stocker des valeurs allant de `Number.MAX_VALUE` (très élevées) à `Number.MIN_VALUE` (très basses). Pour plus d'informations, consultez le *Guide de référence du langage ActionScript 2.0* et la section « Type de données Number », à la page 42.
Object	Complexe. Ce type de données est défini par la classe Object. La classe Object sert de base pour toutes les définitions de classe dans ActionScript. Elle vous permet d'organiser les objets les uns dans les autres (objets imbriqués). Pour plus d'informations, voir « Type de données Object », à la page 43.
String	Primitive. Ce type de données représente une séquence de caractères 16 bits qui peuvent comprendre des lettres, des chiffres et des caractères de ponctuation. Les chaînes sont stockées sous forme de caractères Unicode, au format UTF-16. Toute opération effectuée sur la valeur d'une chaîne renvoie une nouvelle occurrence de la chaîne. Pour plus d'informations, voir « Type de données String », à la page 44.

Type de données	Description
undefined	Primitive. Ce type de données contient une valeur : `undefined`. Il s'agit de la valeur par défaut des occurrences de la classe Object. Seule la valeur `undefined` peut être affectée aux variables qui appartiennent à la classe Object. Pour plus d'informations, voir « Type de données undefined », à la page 45.
Void	Complexe. Ce type de données contient une seule valeur : `void`. Vous l'utilisez pour désigner les fonctions qui ne renvoient aucune valeur. Void est un type de données complexe qui fait référence au type de données Void primitif. Pour plus d'informations, voir « Type de données Void », à la page 45.

Pour un exemple du fichier datatypes.fla, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/DataTypes afin d'accéder à l'exemple.

Type de données Boolean

Une valeur booléenne est soit `true` (vraie), soit `false` (fausse). ActionScript convertit également les valeurs `true` et `false` en 1 et `0` lorsque cela est nécessaire. Les valeurs booléennes sont le plus souvent utilisées dans les instructions ActionScript effectuant des comparaisons pour contrôler le déroulement d'un script.

L'exemple suivant illustre le chargement d'un fichier texte dans un fichier SWF. Il affiche un message dans le panneau de sortie si le chargement du fichier texte échoue ou les paramètres s'il réussit. Pour plus d'informations, consultez les commentaires de l'exemple de code.

```
var my_lv:LoadVars = new LoadVars();
//success est une valeur booléenne
my_lv.onLoad = function(success:Boolean) {
  //si le paramètre success est true, suivre monthNames
  if (success) {
    trace(my_lv.monthNames);
  //si le paramètre success est false, suivre un message
  } else {
    trace("unable to load text file");
  }
};
my_lv.load("http://www.helpexamples.com/flash/params.txt");
```

L'exemple suivant permet de s'assurer que les utilisateurs entrent des valeurs dans deux occurrences de composant TextInput. Deux variables booléennes sont créées, `userNameEntered` et `isPasswordCorrect`. Si ces deux variables renvoient la valeur `true`, un message de bienvenue est affecté à la variable de type chaîne `titleMessage`.

```
// Ajoute deux composants TextInput, un composant Label et
// un composant Button sur la scène.
// Application de typage strict aux trois occurrences de composant.
var userName_ti:mx.controls.TextInput;
var password_ti:mx.controls.TextInput;
var submit_button:mx.controls.Button;
var welcome_lbl:mx.controls.Label;

//Masquer l'étiquette
welcome_lbl.visible = false;

// Crée un objet écouteur utilisé avec le composant Button.
// Lorsque vous cliquez sur le composant Button, le programme recherche
// un nom d'utilisateur et un mot de passe.
var btnListener:Object = new Object();
btnListener.click = function(evt:Object) {
  // Vérifie que l'utilisateur a saisi au moins un caractère dans TextInput
  // occurrences et renvoie une valeur booléenne true/false.
  var userNameEntered:Boolean = (userName_ti.text.length > 0);
  var isPasswordCorrect:Boolean = (password_ti.text == "vertigo");
  if (userNameEntered && isPasswordCorrect) {
    var titleMessage:String = "Welcome " + userName_ti.text + "!";
    welcome_lbl.text = titleMessage;
    //afficher l'étiquette
    welcome_lbl.visible = true;
  }
};
submit_button.addEventListener("click", btnListener);
```

Pour plus d'informations, reportez-vous à « Utilisation des fonctions dans Flash », à la page 185 et à « Présentation des opérateurs logiques », à la page 163.

Type de données MovieClip

Les clips sont des symboles qui peuvent lire des effets animés dans une application Flash. Ils sont le seul type de données faisant référence à un élément graphique. Le type de données MovieClip vous permet de contrôler les symboles de clip au moyen des méthodes de la classe MovieClip.

N'utilisez pas de constructeur pour appeler les méthodes de la classe MovieClip. Vous pouvez créer une instance de clip sur la scène ou créer une instance de façon dynamique. Vous pouvez alors appeler les méthodes de la classe MovieClip avec l'opérateur point (`.`).

Manipulation des clips sur la scène L'exemple suivant appelle les méthodes `startDrag()` et `getURL()` pour différentes occurrences de clip placées sur la scène :

```
my_mc.startDrag(true);
parent_mc.getURL("http://www.adobe.com/support/" + product);
```

Le deuxième exemple renvoie la largeur d'un clip appelé `my_mc` et situé sur la scène. L'instance cible doit être un clip et la valeur renvoyée doit être une valeur numérique.

```
function getMCWidth(target_mc:MovieClip):Number {
  return target_mc._width;
}
trace(getMCWidth(my_mc));
```

Création dynamique de clips La création dynamique de clips à l'aide d'ActionScript évite de devoir les créer manuellement sur la scène ou de les associer à partir de la bibliothèque. Par exemple, vous pouvez créer une galerie d'images à partir d'un grand nombre d'images de vignettes à organiser sur la scène. L'utilisation de `MovieClip.createEmptyMovieClip()` permet de créer une application de bout en bout avec ActionScript.

Pour créer un clip de façon dynamique, utilisez `MovieClip.createEmptyMovieClip()`, comme indiqué dans l'exemple suivant :

```
// Crée un clip parent pour le conteneur.
this.createEmptyMovieClip("image_mc", 9);
// Charge une image dans image_mc.
image_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
```

Le deuxième exemple crée un clip appelé `square_mc` qui utilise l'API de dessin pour dessiner un rectangle. Les gestionnaires d'événement et les méthodes `startDrag()` et `stopDrag()` de la classe MovieClip sont ajoutés pour permettre à l'utilisateur de faire glisser le rectangle.

```
this.createEmptyMovieClip("square_mc", 1);
square_mc.lineStyle(1, 0x000000, 100);
square_mc.beginFill(0xFF0000, 100);
square_mc.moveTo(100, 100);
square_mc.lineTo(200, 100);
square_mc.lineTo(200, 200);
square_mc.lineTo(100, 200);
square_mc.lineTo(100, 100);
square_mc.endFill();
square_mc.onPress = function() {
  this.startDrag();
};
square_mc.onRelease = function() {
  this.stopDrag();
};
```

Pour plus d'informations, consultez le Chapitre 10, « Utilisation des clips », à la page 335 et l'entrée Clip du *Guide de référence du langage ActionScript 2.0*.

Type de données Null

Ce type de données ne peut contenir qu'une valeur, null. Cette valeur signifie en fait *pas de valeur* et symbolise l'absence de données. Vous pouvez affecter la valeur `null` dans de nombreuses situations pour indiquer qu'une propriété ou une variable n'a pas encore reçu de valeur. Par exemple, vous pouvez affecter la valeur `null` dans les situations suivantes :

- Pour indiquer qu'une variable existe, mais n'a pas encore reçu de valeur ;
- Pour indiquer qu'une variable existe, mais ne contient plus de valeur ;
- En tant que valeur de retour d'une fonction, afin d'indiquer qu'aucune valeur n'a pu être retournée par la fonction ;
- En tant que paramètre d'une fonction, afin d'indiquer qu'un paramètre est omis.

Plusieurs méthodes et fonctions renvoient `null` si aucune valeur n'a été définie. L'exemple suivant démontre comment utiliser `null` pour vérifier si les champs du formulaire ont le focus :

```
if (Selection.getFocus() == null) {
  trace("no selection");
}
```

Type de données Number

Ce type numérique correspond à un nombre à virgule flottante à double précision. La valeur minimum d'un objet Number est d'environ 5e-324. Sa valeur maximum est d'environ 1,79E+308.

Vous pouvez manipuler les nombres avec les opérateurs arithmétiques d'addition (+), de soustraction (-), de multiplication (*), de division (/), de modulo (%), d'incrémentation (++) et de décrémentation (--). Pour plus d'informations, voir « Utilisation des opérateurs numériques », à la page 157.

Vous pouvez également utiliser des méthodes des classes intégrées Math et Number pour manipuler les nombres. Pour plus d'informations sur les méthodes et les propriétés de ces classes, consultez les entrées Math et Number du *Guide de référence du langage ActionScript 2.0*.

L'exemple suivant utilise la méthode `sqrt()` (racine carrée) de la classe Math pour renvoyer la racine carrée de 100 :

```
Math.sqrt(100);
```

L'exemple suivant suit un entier compris entre 10 et 17 (inclusivement) :

```
var bottles:Number = 0;
bottles = 10 + Math.floor(Math.random() * 7);
trace("There are " + bottles + " bottles");
```

L'exemple suivant mesure le pourcentage du clip `intro_mc` chargé et le représente sous forme d'entier :

```
var percentLoaded:Number = Math.round((intro_mc.getBytesLoaded() /
  intro_mc.getBytesTotal()) * 100);
```

Type de données Object

Un objet correspond à un ensemble de propriétés. Une *propriété* est un attribut qui décrit l'objet. Par exemple, la transparence d'un objet (tel qu'un clip) est un attribut qui décrit son apparence. Donc, `_alpha` (transparence) est une propriété. Chaque propriété possède un nom et une valeur. La valeur d'une propriété peut être de n'importe quel type de données Flash, même de type Object. Cela vous permet d'arranger les objets les uns dans les autres, ou de les *imbriquer*.

Pour spécifier les objets et leurs propriétés, vous devez utiliser l'opérateur point (`.`). Par exemple, dans le code suivant, `hoursWorked` est une propriété de `weeklyStats`, qui est une propriété de `employee` :

```
employee.weeklyStats.hoursWorked
```

L'objet ActionScript MovieClip possède des méthodes qui vous permettent de contrôler les occurrences de symbole de clip sur la scène. Cet exemple utilise les méthodes `play()` et `nextFrame()` :

```
nomDoccurrenceMC.play();
mc2InstanceName.nextFrame();
```

Vous pouvez aussi créer des objets personnalisés pour organiser les informations dans votre application Flash. Pour ajouter de l'interactivité à votre application avec ActionScript, vous aurez besoin de nombreuses informations : un nom d'utilisateur, son âge et son numéro de téléphone, la vitesse d'une balle, les noms des articles contenus dans un panier d'achat, le nombre d'images chargées ou la dernière touche utilisée sur le clavier, par exemple. La création d'objets personnalisés vous permet d'organiser ces informations dans des groupes, de simplifier la rédaction et de réutiliser vos scripts.

Le code ActionScript suivant affiche un exemple d'utilisation des objets personnalisés pour organiser les informations. Il crée un nouvel objet appelé `user` et trois propriétés, `name`, `age` et `phone`, qui sont de types chaîne et numérique.

```
var user:Object = new Object();
user.name = "Irving";
user.age = 32;
user.phone = "555-1234";
```

Pour plus d'informations, voir « Exemple : Ecriture de classes personnalisées », à la page 238.

Type de données String

Une chaîne est une séquence de caractères (lettres, chiffres et signes de ponctuation, par exemple). Vous insérez des chaînes dans une instruction ActionScript en les plaçant entre des guillemets droits simples (') ou doubles (").

L'un des moyens les plus courants consiste à associer une chaîne à une variable. Par exemple, dans l'instruction suivante, `"L7"` est une chaîne associée à la variable `favoriteBand_str` :

```
var favoriteBand_str:String = "L7";
```

Vous pouvez utiliser l'opérateur d'addition (+) pour *concaténer*, ou réunir, deux chaînes. ActionScript traite les espaces au début ou à la fin d'une chaîne comme faisant partie de la chaîne. L'expression suivante contient un espace après la virgule :

```
var greeting_str:String = "Welcome, " + firstName;
```

Pour inclure un guillemet dans une chaîne, faites-le précéder d'une barre oblique inversée (\). Cette opération s'appelle application d'une *séquence d'échappement* à un caractère. D'autres caractères ne peuvent pas être représentés dans ActionScript sans l'emploi de séquences d'échappement particulières. Le tableau suivant répertorie l'ensemble des caractères d'échappement d'ActionScript :

Séquence d'échappement	Caractère
`\b`	Caractère de retour arrière (ASCII 8)
`\f`	Caractère de changement de page (ASCII 12)
`\n`	Caractère de changement de ligne (ASCII 10)
`\r`	Caractère de retour chariot (ASCII 13)
`\t`	Caractère de tabulation (ASCII 9)
`\"`	Guillemet droit double
`\'`	Guillemet droit simple
`\\`	Barre oblique inverse
`\000 - \377`	Un octet spécifié en octal
`\x00 - \xFF`	Un octet spécifié en hexadécimal
`\u0000 - \uFFFF`	Un caractère Unicode 16 bits spécifié en hexadécimal

Comme sous Java, les chaînes sont inaltérables dans ActionScript. Toute opération modifiant une chaîne renvoie une nouvelle chaîne.

La classe String est une classe intégrée du code ActionScript. Pour plus d'informations sur les méthodes et les propriétés de la classe String, reportez-vous à l'entrée String dans le *Guide de référence du langage ActionScript 2.0*.

Type de données undefined

Ce type de données n'a qu'une seule valeur, `undefined`. Il est affecté automatiquement à une variable qui n'a reçu aucune valeur, que se soit du code ou de l'interaction de l'utilisateur.

La valeur `undefined` est affectée automatiquement. Contrairement à `null`, vous ne pouvez pas l'affecter à une variable ou une propriété. Ce type de données permet de vérifier si la variable est définie ou non. Vous pouvez alors écrire du code qui ne s'exécute que lorsque l'application est en cours d'exécution, comme indiqué dans l'exemple suivant :

```
if (init == undefined) {
  trace("initializing app");
  init = true;
}
```

Si votre application comporte plusieurs images, le code ne s'exécute pas une deuxième fois parce que la variable `init` n'est plus « undefined ».

Type de données Void

Ce type de données n'a qu'une seule valeur, `void`. Il est utilisé dans la définition des fonctions pour indiquer qu'elles ne renvoient pas de valeurs, comme indiqué dans l'exemple suivant :

```
//Crée une fonction renvoyant le type Void
function displayFromURL(url:String):Void {}
```

Affectation des types de données et typage strict

Dans Flash, les variables vous permettent de stocker des valeurs dans votre code. Vous pouvez déclarer explicitement le type d'objet d'une variable lorsque vous la créez. Cette opération est appelée le *typage strict*.

Si vous ne déclarez pas de façon explicite qu'un élément contient un nombre, une chaîne ou tout autre type de donnée, lors de l'exécution du script Flash Player tente d'identifier le type de données de cet élément lorsqu'il est affecté. Si vous affectez une valeur à une variable comme indiqué dans l'exemple suivant, lors de l'exécution du script, Flash Player évalue l'élément situé à droite de l'opérateur et détermine qu'il est de type numérique :

```
var x = 3;
```

Dans la mesure où `x` n'a pas été déclaré par typage strict, le compilateur ne peut pas déterminer son type. Ainsi, pour le compilateur, la variable *x* peut être de n'importe quel type. (Voir la section « Affectation d'un type de données », à la page 47.) Une affectation ultérieure pourra changer le type de `x`. Par exemple, l'instruction `x = "hello"` change le type de `x` en String.

ActionScript convertit toujours les types de données primitifs (tels que Boolean, Number, String, null ou undefined) automatiquement lorsqu'une expression en a besoin et que les variables ne sont pas déclarées par typage strict.

Le typage strict des données offre plusieurs avantages lors de la compilation. La déclaration des types de données (typage strict) permet de prévenir et de diagnostiquer les erreurs de votre code au moment de sa compilation. Pour déclarer une variable par typage strict, utilisez le format suivant :

```
var variableName:datatype;
```

REMARQUE

Le typage strict des variables est parfois appelé *typage fort*.

Comme les incompatibilités de type de données déclenchent des erreurs de compilation, le typage strict facilite le débogage lors de la compilation et permet donc d'éviter d'affecter un type de données incorrect à une variable existante. Lors de la programmation, le typage strict des données active les conseils de code dans l'éditeur ActionScript (mais vous devez toujours utiliser les suffixes de nom d'occurrence pour les éléments visuels).

L'utilisation du typage strict vous empêche d'affecter, par inadvertance, un type de valeur incorrect à une variable. Flash vérifie les erreurs de typage lors de la compilation. En cas de type de valeur incorrect, il affiche un message d'erreur. Ainsi, l'application du typage strict permet de s'assurer que vous ne tentez pas d'accéder à des propriétés ou des méthodes qui ne font pas partie d'un type d'objet. Le typage strict permet également de bénéficier automatiquement des conseils de code proposés par l'éditeur d'ActionScript pour les objets.

Pour plus d'informations sur la création de variables, consultez la section « Présentation des variables », à la page 51. Pour plus d'informations sur l'appellation des variables, consultez la section « A propos de l'appellation des variables », à la page 57. Pour plus d'informations sur l'affectation des types de données et sur les types appropriés, consultez la section « Affectation d'un type de données », à la page 47.

Pour un exemple du fichier datatypes.fla, qui vous explique comment utiliser des types de données dans une application, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/DataTypes afin d'accéder à l'exemple.

Affectation d'un type de données

Vous devez affecter des types de données chaque fois que vous définissez une variable, que vous la déclariez à l'aide d'un mot-clé `var`, que vous créiez un argument de fonction ou que vous définissiez un type de retour de fonction ou une variable à utiliser dans une boucle `for` ou `for..in`. Pour affecter un type de données, utilisez la *syntaxe à deux points* : le nom de la variable est suivi de deux points, puis du type de données :

```
var my_mc:MovieClip;
```

Les types de données offrent de nombreuses possibilités : des types natifs, tels que Number, String, Boolean, jusqu'aux classes intégrées de Flash Player 8, telles que BitmapData, FileReference ou même les classes personnalisées écrites par vos soins ou par d'autres développeurs. Les types de données les plus courants que vous devrez définir sont les types intégrés, tels que Number, String, Boolean, Array ou Object qui apparaissent dans les exemples de code suivants.

Pour affecter un type de données spécifique à un élément, spécifiez son type à l'aide d'une syntaxe utilisant le mot-clé `var` ainsi que deux points, comme illustré ci-dessous :

```
// Typage strict de variable ou d'objet
var myNum:Number = 7;
var birthday:Date = new Date();

// Typage strict de paramètres
function welcome(firstName:String, age:Number) {
}

// Typage strict de paramètre et de valeur renvoyée
function square(myNum:Number):Number {
  var squared:Number = myNum * myNum;
  return squared;
}
```

Vous pouvez déclarer le type des objets en fonction des classes intégrées (Button, Date, etc.) et des classes et interfaces que vous créez. Dans l'exemple suivant, si vous avez un fichier qui s'appelle Student.as dans lequel vous avez défini une classe Student, vous pouvez spécifier que les objets que vous créez sont de type Student :

```
var myStudent:Student = new Student();
```

Par exemple, supposons que vous tapiez le code suivant :

```
// dans le fichier de classe Student.as
class Student {
  public var status:Boolean; // Propriété des objets Student
}

// Dans le fichier FLA
var studentMaryLago:Student = new Student();
studentMaryLago.status = "enrolled"; /* Incompatibilité de types dans
  l'instruction d'affectation : type String présent au lieu du type
  Boolean. */
```

Lorsque Flash compile ce script, une erreur d'incompatibilité de types est générée, car le fichier SWF s'attend à une valeur booléenne.

Si vous écrivez une fonction sans type de renvoi, vous pouvez déclarer le type de renvoi Void pour cette fonction. Ou, si vous créez un raccourci vers une fonction, vous pouvez affecter le type de données Function à la nouvelle variable. Pour indiquer le type Function ou Void pour les objets, inspirez-vous de l'exemple suivant :

```
function sayHello(name_str:String):Void {
  trace("Hello, " + name_str);
}
sayHello("world"); // Hello, world
var greeting:Function = sayHello;
greeting("Augustus"); // Hello, Augustus
```

Un des autres avantages du typage strict des données réside dans le fait que Flash affiche automatiquement des conseils de code pour les objets intégrés que vous déclarez par typage stricte. Pour plus d'informations, voir « Affectation des types de données et typage strict », à la page 45.

Les fichiers publiés avec ActionScript 1.0 ne respectent pas le typage strict des données lors de la compilation. Par conséquent, l'attribution d'un type de valeur incorrect à une variable que vous avez typée de façon stricte ne génère pas d'erreur de compilation.

```
var myNum:String = "abc";
myNum = 12;
/* Aucune erreur dans ActionScript 1.0, mais incompatibilité dans
  ActionScript 2.0 */
```

Ceci est dû au fait que lorsque vous publiez un fichier pour ActionScript 1.0, Flash interprète une instruction telle que `var myNum:String = "abc"` comme une syntaxe à barre oblique et non comme un typage strict. (ActionScript 2.0 ne prend pas en charge la syntaxe à barre oblique.) Il peut alors en résulter un objet affecté à une variable de type erroné.
Le compilateur ignore alors les appels de méthodes non valides et les références de propriété non définies.

Les fichiers publiés à l'aide d'ActionScript 2.0 peuvent utiliser le typage des données en option. Ainsi, si vous implémentez le typage strict dans votre code, assurez-vous de définir vos paramètres de publication sur ActionScript 2.0. Vous pouvez définir les paramètres de publication, ainsi que la version d'ActionScript à utiliser pour la publication de vos fichiers, soit à partir du menu principal (Fichier > Paramètres de publication), soit en cliquant sur le bouton Paramètres dans l'inspecteur des propriétés (assurez-vous qu'aucune occurrence n'est sélectionnée). Pour utiliser une version spéciale d'ActionScript ou de Flash Player, ouvrez l'onglet Flash dans la boîte de dialogue Paramètres de publication, puis sélectionnez la version ActionScript désirée dans le menu contextuel.

Pour plus d'informations sur la vérification des types de données, consultez la section « Vérification du type de données », à la page 49.

Vérification du type de données

Cette opération consiste à vérifier que le type d'une variable est compatible avec une expression. Flash vérifie donc que le type que vous indiquez pour une variable convient aux valeurs que vous affectez à celle-ci. Pour plus d'informations sur le typage strict et l'affectation des types de données, consultez les sections « Affectation des types de données et typage strict », à la page 45 et « Affectation d'un type de données », à la page 47.

La vérification des types peut être effectuée lors de la compilation ou de l'exécution. Si vous utilisez le typage strict des données, le type est vérifié lors de la compilation. ActionScript étant un langage typé dynamiquement, ActionScript peut également vérifier le type à l'exécution.

Par exemple, le code suivant n'indique aucun type de données pour le paramètre `xParam`. A l'exécution, vous utilisez ce paramètre pour stocker une valeur numérique `Number`, puis une valeur de chaîne de caractères `String`. La fonction `dynamicTest()` utilise ensuite l'opérateur `typeof` pour connaître le type du paramètre, String ou Number.

```
function dynamicTest(xParam) {
  if (typeof(xParam) == "string") {
    var myStr:String = xParam;
    trace("String: " + myStr);
  } else if (typeof(xParam) == "number") {
    var myNum:Number = xParam;
    trace("Number: " + myNum);
  }
}
dynamicTest(100);
dynamicTest("one hundred");
```

Vous n'avez pas besoin de déclarer explicitement le type de ces données dans votre code ActionScript. Le compilateur d'ActionScript vous permet d'utiliser des propriétés et d'invoquer des méthodes qui n'existent pas au moment de la compilation. Ainsi, vous pouvez créer des propriétés ou affecter des méthodes dynamiquement au moment de l'exécution.

La flexibilité offerte par la vérification dynamique du type de données implique l'utilisation de propriétés et de méthodes inconnues au moment de la compilation. Le code étant moins restrictif, cette flexibilité est avantageuse dans certains cas de programmation. Par exemple, le code suivant crée une fonction nommée `runtimeTest()` qui appelle une méthode et renvoie une propriété, toutes deux inconnues du compilateur. Ce code ne générera aucune erreur lors de la compilation. Cependant, si la propriété ou la méthode est inaccessible à l'exécution, une erreur se produira à ce moment-là.

```
function runtimeTest(myParam) {
  myParam.someMethod();
  return myParam.someProperty;
}
```

Identification du type de données

Lors du test et du débogage de vos programmes, des problèmes liés aux types de données peuvent se produire. Ou, si vous utilisez des variables non explicitement associées à un type de données, il sera plus pratique de connaître le type d'une variable donnée. Avec ActionScript, vous pouvez identifier le type de données d'un élément. Vous pouvez utiliser l'opérateur `typeof` pour récupérer des informations sur les données.

Cependant, n'oubliez pas que cet opérateur ne renvoie pas d'informations sur la classe d'une occurrence.

L'exemple suivant présente l'utilisation de l'opérateur `typeof` pour renvoyer la nature de l'objet à identifier :

```
// Crée une nouvelle occurrence de la classe LoadVars.
var my_lv:LoadVars = new LoadVars();

/* L'opérateur typeof n'indique pas la classe, il spécifie uniquement que
  myLV est un objet. */
var typeResult:String = typeof(my_lv);
trace(typeResult); // Objet
```

Dans cet exemple, vous créez une nouvelle variable String nommée `myName`, puis vous la convertissez en type Number :

```
var myName:String = new String("17");
trace(myName instanceof String); // true
var myNumber:Number = new Number(myName);
trace(myNumber instanceof Number); // true
```

Pour plus d'informations sur ces opérateurs, reportez-vous à l'`opérateur typeof` et l'`opérateur instanceof` dans le *Guide de référence du langage ActionScript 2.0*. Pour plus d'informations sur le test et le débogage, consultez le guide *Utilisation de Flash*. Pour plus d'informations sur les héritages et les interfaces, consultez le Chapitre 7, « Héritage », à la page 281. Pour plus d'informations sur les classes, consultez le Chapitre 6, « Classes », à la page 197.

Présentation des variables

Une *variable* est un conteneur qui stocke des informations. Le code suivant montre une variable dans ActionScript :

```
var myVariable:Number = 10;
```

Cette variable contient une valeur numérique. L'utilisation de `:Number` dans le code précédent affecte le type de valeur que la variable contient, opération appelée *typage des données*. Pour plus d'informations sur le typage des données, consultez les sections « Affectation des types de données et typage strict », à la page 45 et « Affectation d'un type de données », à la page 47.

Le conteneur (représenté par le nom de variable) est toujours le même dans tout votre code ActionScript, mais son contenu (sa *valeur*) peut évoluer. Vous pouvez changer plusieurs fois la valeur d'une variable dans un script. La modification de la valeur d'une variable pendant la lecture du fichier SWF permet d'enregistrer les informations relatives aux actions de l'utilisateur, d'enregistrer les valeurs modifiées pendant la lecture du fichier SWF ou d'évaluer si une condition est vraie ou fausse (`true` ou `false`). La variable devra peut-être être continuellement mise à jour pendant la lecture du fichier SWF, par exemple lorsque le score d'un joueur change dans un jeu Flash. Les variables sont indispensables pour créer et gérer les interactions de l'utilisateur dans un fichier SWF.

Il est toujours judicieux d'affecter une valeur à une variable lors de sa première déclaration. L'affectation d'une valeur initiale est appelée *initialisation* de la variable. Cette opération est souvent effectuée sur l'Image 1 du scénario ou depuis la classe qui est chargée au début de la lecture du fichier SWF. Il existe différentes sorte de variables qui sont affectées par domaine. Pour plus d'informations sur les différentes sortes de variables et sur les domaines, consultez la section « Variables et domaine », à la page 62.

CONSEIL	L'initialisation d'une variable facilite le suivi et la comparaison de sa valeur pendant la lecture du fichier SWF.

REMARQUE	Flash Player 7 et ses versions ultérieures n'évaluent pas les variables non initialisées de la même manière que Flash Player 6 et ses versions précédentes. Si vous avez écrit des scripts pour Flash Player 6 et prévoyez d'écrire ou porter des scripts pour Flash Player 7 ou une version plus récente, vous devez comprendre ces différences pour éviter un comportement inattendu.

Les variables peuvent contenir différents types de données. Pour plus d'informations, consultez la section « Présentation des types de données », à la page 36. Le type de données d'une variable influence la façon dont sa valeur est modifiée par un script.

Les types d'informations le plus souvent stockés dans une variable sont les URL (String), les noms d'utilisateur (String), les résultats d'opérations mathématiques (Number), le nombre d'occurrences d'un événement (Number) ou si un bouton a été actionné (Boolean). Chaque fichier SWF et chaque occurrence d'objet (tel qu'un clip) dispose d'un jeu de variables, dont la valeur est indépendante des variables figurant dans d'autres fichiers SWF ou clips.

Pour tester la valeur d'une variable, utilisez l'instruction `trace()` pour envoyer la valeur au panneau de sortie. La valeur apparaît ensuite dans le panneau de sortie lorsque vous testez le fichier SWF dans l'environnement de test. Par exemple, `trace(hoursWorked)` envoie la valeur de la variable `hoursWorked` au panneau de sortie dans l'environnement de test. Vous pouvez également vérifier et définir les valeurs des variables avec le débogueur dans l'environnement de test.

Pour plus d'informations sur les variables, consultez les sections suivantes :

- « Déclaration des variables », à la page 53
- « Affectation des valeurs », à la page 53
- « A propos de l'appellation des variables », à la page 57
- « Utilisation des variables dans une application », à la page 58
- « Variables et domaine », à la page 62
- « A propos des valeurs par défaut », à la page 53

Déclaration des variables

Vous pouvez déclarer les variables sur une image du scénario, directement dans un objet ou dans un fichier de classe externe.

Définissez les variables à l'aide du mot-clé `var` et respectez leurs conventions d'appellation. Comme dans l'exemple suivant, vous pouvez déclarer une variable nommée `firstName` :

```
var firstName:String;
```

Lorsque vous déclarez une variable, vous lui affectez un type de données. Dans le cas présent, vous affectez le type String à la variable `firstName`. Pour plus d'informations sur l'affectation du type de données, consultez le « Affectation des types de données et typage strict », à la page 45.

A propos des valeurs par défaut

La *valeur par défaut* est le contenu d'une variable avant que vous définissiez sa valeur. Vous *initialisez* une variable lorsque vous lui affectez sa première valeur. Si vous déclarez une variable sans définir sa valeur, cette variable est *non initialisée*. La valeur par défaut d'une variable non initialisée est `undefined`. Pour plus d'informations sur la création et l'utilisation des variables, consultez la section « Présentation des variables », à la page 51.

Affectation des valeurs

Vous pouvez définir une *valeur* comme contenu actuel d'une variable. Cette valeur peut être composée de chaînes, nombres, tableaux, objets, XML, dates ou même de classes personnalisées que vous créez. N'oubliez pas que vous déclarez les variables dans Flash à l'aide du mot-clé `var`. Lorsque vous déclarez une variable, vous lui affectez également un type de données. Vous pouvez également lui attribuer une valeur tant que celle-ci correspond au type de données affecté à la variable.

L'exemple suivant présente la création d'une variable nommée `catName` :

```
var catName:String;
```

Après la déclaration de la variable, vous pouvez lui affecter une valeur. La ligne de code ActionScript précédente doit être suivie de cette ligne :

```
catName = "Pirate Eye";
```

REMARQUE

`Pirate Eye` étant une chaîne (String), sa valeur doit être entre guillemets.

Cet exemple affecte la valeur de `Pirate Eye` à la variable `catName`. Lorsque vous déclarez la variable, vous pouvez également lui attribuer une valeur immédiatement et non ultérieurement comme dans les exemples précédents. Vous pouvez définir la variable `catName` lors de sa déclaration, comme dans l'exemple suivant :

```
var catName:String = "Pirate Eye";
```

Si vous souhaitez afficher la valeur de la variable `catName` dans l'environnement de test, utilisez l'instruction `trace()`. Cette instruction envoie la valeur au panneau de sortie. Vous pouvez suivre l'évolution de la valeur de la variable `catName` et vérifier si les guillemets sont absents dans son contenu actuel à l'aide du code ActionScript suivant :

```
var catName:String = "Pirate Eye";
trace(catName); // Pirate Eye
```

N'oubliez pas que la valeur affectée doit être compatible avec le type de données déclaré (String dans le cas présent). Si par la suite vous tentez d'affecter un nombre à la variable `catName`, par exemple`catName = 10`, l'erreur suivante apparaîtra dans le panneau de sortie lors du test du fichier SWF :

```
Type mismatch in assignment statement: found Number where String is
  required.
```

Cette erreur indique que vous avez tenté d'affecter un type de données incorrect à une variable définie.

Lorsque vous affectez une valeur numérique à une variable, les guillemets ne sont pas nécessaires, comme le montre le code suivant :

```
var numWrinkles:Number = 55;
```

Pour modifier la valeur de `numWrinkles` ultérieurement dans votre code, affectez une nouvelle valeur à l'aide du code ActionScript suivant :

```
numWrinkles = 60;
```

Lorsque vous réaffectez la valeur d'une variable existante, il n'est pas nécessaire d'utiliser le mot-clé `var` ni de définir le type de données de la variable (dans le cas présent, `Number`).

Si la valeur est numérique ou booléenne (`true` ou `false`), les guillemets sont inutiles. Des exemples de valeurs numériques et booléennes sont présentés dans le fragment de code suivant :

```
var age:Number = 38;
var married:Boolean = true;
var hasChildren:Boolean = false;
```

Dans l'exemple précédent, la variable `age` contient la valeur d'un entier (non décimal), bien que vous puissiez utiliser une valeur décimale ou à virgule flottante telle que 38,4. Les variables booléennes (telles que `married` ou `hasChildren`) ont deux valeurs possibles, `true` et `false`.

Si vous souhaitez créer un tableau et lui affecter des valeurs, le format est légèrement différent, comme dans le code suivant :

```
var childrenArr:Array = new Array("Pylon", "Smithers", "Gil");
```

Il existe une autre syntaxe (abrégée) pour créer un tableau à l'aide des opérateurs d'accès tableau, qui utilisent les parenthèses (`[]`). Vous pouvez réécrire l'exemple précédent comme suit :

```
var childrenArr:Array = ["Pylon", "Smithers", "Gil"];
```

Pour plus d'informations sur la création de tableaux et leurs opérateurs d'accès, consultez les sections « Présentation des tableaux », à la page 132 et « Utilisation de la syntaxe à point pour cibler une occurrence », à la page 86.

De même, vous pouvez créer un nouvel objet appelé `myObj`. Pour créer un objet, utilisez l'une des méthodes suivantes. La première méthode (et la plus longue) pour créer un tableau avec du code est la suivante :

```
var myObj:Object = new Object();
myObj.firstName = "Steve";
myObj.age = 50;
myObj.childrenArr = new Array("Mike", "Robbie", "Chip");
```

La seconde méthode (la plus courte) pour créer le tableau `myObj` avec du code est la suivante :

```
var myObj:Object = {firstName:"Steve", age:50, childrenArr:["Mike",
  "Robbie", "Chip"]};
```

Comme le montre cet exemple, le choix de la méthode courte permet d'économiser beaucoup de saisie et de temps, particulièrement lorsque vous définissez des occurrences d'objets. Il est important de bien connaître cette syntaxe de rechange car vous la rencontrerez en travaillant avec d'autres équipes ou dans du code ActionScript écrit par des tiers sur Internet ou dans les livres spécialisés.

REMARQUE	Toutes les variables n'ont pas besoin d'être définies explicitement. Certaines variables sont créées automatiquement par Flash pour vous. Par exemple, pour connaître les dimensions de la scène, vous pouvez utiliser les valeurs des deux variables prédéfinies suivantes : `Stage.width` et `Stage.height`.

A propos des opérateurs et des variables

Les symboles mathématiques présents dans le code peuvent prêter à confusion.
Dans ActionScript, ces symboles sont appelés des *opérateurs*. Les opérateurs calculent une nouvelle valeur à partir d'une ou plusieurs valeurs et vous les utilisez pour affecter une valeur à une variable dans votre code. Vous utilisez l'opérateur d'égalité (=) pour affecter une valeur à une variable :

```
var username:String = "Gus";
```

L'opérateur d'addition (+) est un autre exemple, utilisé pour ajouter des valeurs numériques entre elles et produire ainsi une nouvelle valeur. Si vous utilisez l'opérateur + avec des valeurs de chaînes (string), les chaînes seront concaténées. Les valeurs manipulées avec des opérateurs sont appelées *opérandes*.

Pour affecter une valeur à une variable, vous utilisez un opérateur. Par exemple, le script suivant utilise l'opérateur d'affectation pour donner la valeur 7 à la variable `numChildren` :

```
var numChildren:Number = 7;
```

Si vous devez modifier la valeur de la variable `numChildren`, utilisez le code suivant :

```
numChildren = 8;
```

REMARQUE	Le mot-clé `var` n'est pas nécessaire car la variable a déjà été définie.

Pour plus d'informations sur l'utilisation des opérateurs dans le code ActionScript, consultez la section « Présentation des opérateurs », à la page 145.

A propos de l'appellation des variables

Prenez soin de bien choisir les noms que vous attribuez aux variables car, bien que leur choix soit entièrement libre, certaines règles doivent être respectées. Le nom des variables doit suivre les règles suivantes :

- Toute variable doit être un identifiant.

> **REMARQUE** Les *identifiants* sont des noms de variable, de propriété, d'objet, de fonction ou de méthode. Le premier caractère des identifiants doit être une lettre, un trait de soulignement (_) ou un signe dollar ($). Chaque caractère qui suit peut être une lettre, un chiffre, un trait de soulignement ou un dollar.

- Une variable ne peut pas être un mot-clé, ni un littéral ActionScript, tel que `true`, `false`, `null` ou `undefined`. Pour plus d'informations sur les littéraux, consultez la section « Présentation des littéraux », à la page 99.
- Toute variable doit être unique dans son domaine (consultez la section « Variables et domaine », à la page 62).
- Le nom d'une variable ne doit pas correspondre à un élément du langage ActionScript, tel qu'un nom de classe.

Si vous ne respectez pas ces règles lors de l'attribution des noms de variables, des erreurs de syntaxe ou des résultats inattendus risquent de se produire. Dans l'exemple suivant, si vous nommez une variable `new`, lorsque vous testez votre document, Flash générera une erreur de compilation :

```
// Ce code fonctionne comme prévu.
var helloStr:String = new String();
trace(helloStr.length); // 0
// Mais si une variable porte le même nom qu'une classe intégrée....
var new:String = "hello"; // erreur : identifiant attendu
var helloStr:String = new String();
trace(helloStr.length); // non défini
```

L'éditeur ActionScript prend en charge les conseils de code pour les classes intégrées et pour les variables basées sur ces classes. Si vous souhaitez obtenir des conseils de code pour un type d'objet particulier que vous avez affecté à une variable, vous pouvez définir strictement le type de cette dernière. Les conseils de code fournissent des astuces de syntaxe sous forme d'info-bulles et un menu contextuel qui accélère votre rédaction du code.

Par exemple, vous tapez le code suivant :

```
var members:Array = new Array();
members.
```

Dès que vous tapez le point (.) dans le panneau Actions, Flash affiche la liste des méthodes et propriétés disponibles pour les objets Array.

Pour connaître les conventions de codage pour l'appellation des variables, consultez la section « Appellation des variables », à la page 720.

Utilisation des variables dans une application

Dans cette section, vous utilisez des variables dans des fragments de code ActionScript. Vous devez déclarer et initialiser une variable dans un script avant de pouvoir l'utiliser dans une expression. Les expressions sont des combinaisons d'opérandes et d'opérateurs qui représentent une valeur. Par exemple, dans l'expression `i+2`, `i` et `2` sont des opérandes et `+` est un opérateur.

Si vous n'initialisez pas une variable avant de l'utiliser dans une expression, elle reste indéfinie et risque de provoquer des résultats inattendus. Pour plus d'informations sur la syntaxe des expressions, consultez le Chapitre 4, « Eléments fondamentaux du langage et de la syntaxe », à la page 81.

Si vous utilisez une variable indéfinie comme dans l'exemple suivant, elle prend la valeur `NaN` dans Flash Player 7 et les versions ultérieures, et votre script est susceptible de générer des résultats inattendus :

```
var squared:Number = myNum * myNum;
trace(squared); // NaN
var myNum:Number = 6;
```

Dans l'exemple suivant, l'instruction déclarant et initialisant la variable `myNum` est placée en premier, pour que `squared` puisse être remplacé par une valeur :

```
var myNum:Number = 6;
var squared:Number = myNum * myNum;
trace(squared); // 36
```

La même chose se produit lorsque vous transmettez une valeur non définie à une méthode ou à une fonction, comme dans l'exemple suivant.

Comparaison de variables définies et non définies transférées à une fonction

1. Faites glisser un composant Button du panneau Composants sur la scène.
2. Ouvrez l'inspecteur des propriétés et tapez **bad_button** dans la zone de texte Nom de l'occurrence.
3. Tapez le code suivant dans l'image 1 du scénario.

```
// Ne fonctionne pas
function badClickListener(evt:Object):Void {
  getURL(targetUrl);
  var targetUrl:String = "http://www.adobe.com";
}
bad_button.addEventListener("click", badClickListener);
```

4. Sélectionnez Contrôle > Tester l'animation, et remarquez que le bouton ne fonctionne pas (la page Web ne s'ouvre pas).
5. Faites glisser un composant Button sur la scène. Sélectionnez le bouton.

6. Ouvrez l'inspecteur des propriétés et tapez **good_button** dans le champ de texte Nom de l'occurrence.
7. Ajoutez le code ActionScript suivant à l'image 1 du scénario (à la suite du précédent code ActionScript ajouté) :

```
// Fonctionne
function goodClickListener(evt:Object):Void {
  var targetUrl:String = "http://www.adobe.com";
  getURL(targetUrl);
}
good_button.addEventListener("click", goodClickListener);
```

8. Sélectionnez Contrôle > Tester l'animation, puis cliquez sur le second bouton ajouté à la scène.

   Ce bouton fonctionne correctement et ouvre la page Web.

Le type de données contenu dans une variable affecte les conditions et le moment où sa valeur sera modifiee. Les types de données primitifs, tels que les chaînes et les chiffres, sont *transmis par valeur* : la valeur actuelle de la variable est reprise, et non pas une référence à cette valeur. Parmi les exemples de type de données complexes, on trouve Array et Object.

Dans l'exemple suivant, vous définissez `myNum` sur `15` et copiez la valeur dans `otherNum`. Lorsque vous modifiez `myNum` en 30 (à la ligne 3 du code), la valeur de `otherNum` demeure `15` car `otherNum` ne regarde pas dans `myNum` pour connaître sa valeur. La variable `otherNum` contient la valeur de `myNum` qu'elle a reçue (à la ligne 2 du code).

Utilisation des variables dans votre code ActionScript

1. Créez un document Flash, puis enregistrez-le sous le nom de **var_example.fla**.
2. Sélectionnez l'image 1 du scénario, puis tapez le code suivant dans le panneau Actions :

```
var myNum:Number = 15;
var otherNum:Number = myNum;
myNum = 30;
trace(myNum); // 30
trace(otherNum); // 15
```

   Lorsque vous modifiez `myNum` en 30 (à la ligne 3 du code), la valeur de `otherNum` demeure `15` car `otherNum` ne regarde pas dans `myNum` pour connaître sa valeur. La variable `otherNum` contient la valeur de `myNum` qu'elle a reçue (à la ligne 2 du code).
3. Choisissez Contrôle > Tester l'animation pour voir les valeurs affichées dans le panneau de sortie.

4. Maintenant, ajoutez le code ActionScript suivant après celui ajouté à l'étape 2 :

```
function sqr(myNum:Number):Number {
  myNum *= myNum;
  return myNum;
}
var inValue:Number = 3;
var outValue:Number = sqr(inValue);
trace(inValue); // 3
trace(outValue); // 9
```

   Dans ce code, la variable `inValue` contient une valeur primitive, `3`, la valeur est donc transmise à la fonction `sqr()` et la valeur renvoyée est `9`. La valeur de la variable `inValue` ne change pas, bien que la valeur de `myNum` change dans la fonction.

5. Choisissez Contrôle > Tester l'animation pour voir les valeurs affichées dans le panneau de sortie.

Le type de données Object peut contenir tant d'informations complexes qu'une variable de ce type ne contient pas de valeur réelle, mais une référence à la valeur. Cette référence est un alias qui désigne le contenu de la variable. Lorsque la variable a besoin de connaître sa valeur, la référence demande le contenu et renvoie la réponse sans transférer la valeur à la variable.

Pour plus d'informations sur le transfert d'une variable par référence, consultez la section « Transfert d'une variable par référence », à la page 60.

Transfert d'une variable par référence

Les types de données Array et Object contenant une référence à une valeur au lieu de leur valeur réelle, soyez prudent lorsque vous travaillez avec des tableaux et des objets.

L'exemple suivant montre comment transmettre un objet par référence. Lorsque vous créez une copie du tableau, vous créez en réalité une copie de la référence (ou *alias*) au contenu du tableau. Lorsque vous modifiez le contenu du second tableau, vous modifiez à la fois le contenu du premier et du second tableau car ils pointent tous les deux vers la même valeur.

Transmission d'un objet par référence

1. Sélectionnez Fichier > Nouveau puis Document Flash pour créer un fichier FLA et l'enregistrer sous le nom de **copybyref.fla**.
2. Sélectionnez l'image 1 du scénario, puis tapez le code suivant dans le panneau Actions :

```
var myArray:Array = new Array("tom", "josie");
var newArray:Array = myArray;
myArray[1] = "jack";
trace(myArray); // tom,jack
trace(newArray); // tom,jack
```

3. Choisissez Contrôle > Tester l'animation pour tester le code ActionScript.

   Ce code crée un objet Array appelé `myArray` qui contient deux éléments. Vous créez la variable `newArray` et transmettez une référence à `myArray`. Lorsque vous modifiez le deuxième élément de `myArray` en `jack`, cela affecte toutes les variables qui y font référence. L'instruction `trace()` envoie `tom,jack` au panneau Sortie.

REMARQUE	Flash utilise un index basé sur zéro, ce qui signifie que 0 est le premier élément du tableau, 1 le deuxième, etc.

Dans l'exemple suivant, `myArray` contient un objet Array que vous transmettez à la fonction `zeroArray()` par référence. La fonction `tableauNul()` accepte un objet Array comme paramètre et définit tous les éléments de ce tableau sur 0. Elle peut modifier ce tableau car il est transmis par référence.

Transmission d'un tableau par référence

1. Sélectionnez Fichier > Nouveau puis Document Flash pour créer un fichier FLA et l'enregistrer sous le nom de **arraybyref.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
function zeroArray (theArr:Array):Void {
  var i:Number;
  for (i = 0; i < theArr.length; i++) {
    theArr[i] = 0;
  }
}

var myArr:Array = new Array();
myArr[0] = 1;
myArr[1] = 2;
myArr[2] = 3;
trace(myArr); // 1,2,3
zeroArray(myArr);
trace(myArr); // 0,0,0
```

3. Choisissez Contrôle > Tester l'animation pour tester votre code ActionScript.

   La première instruction `trace()` de ce code ActionScript affiche le contenu original du tableau `myArray` (`1,2,3`). Après votre appel de la fonction `zeroArray()` et votre transfert d'une référence au tableau`myArray`, toutes les valeurs du tableau sont écrasées et définies sur zéro. La prochaine instruction `trace()` affiche le nouveau contenu original du tableau `myArray` (`0,0,0`). Le tableau ayant été transmis par référence et non par valeur, il est inutile de renvoyer le contenu actualisé du tableau provenant de la fonction `zeroArray()`.

Pour plus d'informations sur les tableaux, consultez la section « Présentation des tableaux », à la page 132.

Variables et domaine

Le domaine d'une variable fait référence au domaine dans lequel la variable est connue (*définie*) et peut être référencée. Il peut s'agir d'un certain scénario ou d'une fonction, ou d'une application entière. Pour plus d'informations sur le domaine, consultez la section « Domaine et ciblage », à la page 91.

Lorsque vous développez des applications Flash avec ActionScript, la compréhension du domaine des variables est capitale. Le domaine indique non seulement où et quand vous pouvez faire référence aux variables, mais également la durée d'existence de chaque variable dans l'application. Lorsque vous définissez des variables dans le corps d'une fonction, elles cessent d'exister dès que la fonction spécifiée se termine. Si vous tentez de faire référence à des objets dans un domaine incorrect ou à des variables qui ont expiré, vous obtenez des erreurs dans vos documents Flash, ce qui entraîne des comportements inattendus ou des défaillances de fonctionnalité.

Il existe trois types de domaine de variable dans ActionScript :

- Variables globales et fonctions sont visibles par tout scénario et domaine du document. Toute variable globale est donc définie dans chaque zone de votre code.
- Variables de scénario sont disponibles pour tous les scripts de ce scénario.
- Variables locales sont disponibles dans le corps de la fonction dans lequel elles sont déclarées (délimité par des accolades). Les variables locales sont donc définies dans une seule partie de votre code.

Pour consulter les recommandations sur l'utilisation des domaines et des variables, consultez le Chapitre 4, « Domaine et ciblage », à la page 91.

REMARQUE

Les classes ActionScript 2.0 que vous créez prennent en charge les domaines de variables publics, privés et statiques. Pour plus d'informations, voir « Présentation des membres de classe », à la page 224 et « Contrôle de l'accès des membres dans vos classes », à la page 249.

Vous ne pouvez pas typer les variables globales de façon stricte. Pour obtenir plus d'informations et une solution de rechange, consultez la section « Variables globales », à la page 63.

Variables globales

Les variables et les fonctions globales sont visibles par tout scénario et domaine du document. Pour déclarer (ou *créer*) une variable globale, faites précéder son nom de l'identifiant `_global` et n'utilisez pas la syntaxe `var =`. Par exemple, le code suivant crée la variable globale `myName` :

```
var _global.myName = "George"; // Syntaxe incorrecte pour la variable
                               // globale
_global.myName = "George"; // Syntaxe correcte pour la variable globale
```

Toutefois, si vous initialisez une variable locale portant le même nom qu'une variable globale, vous n'avez pas accès à cette dernière dans le domaine de la variable locale, comme indiqué dans l'exemple suivant :

```
_global.counter = 100; // Déclare une variable globale
trace(counter); // Accède à la variable globale et affiche 100
function count():Void {
  for (var counter:Number = 0; counter <= 2; counter++) { // Variable
                                                          // locale
    trace(counter); // Accède à la variable locale et affiche de 0 à 2
  }
}
count();
trace(counter); // Accède à la variable globale et affiche 100
```

Cet exemple indique uniquement que la variable globale n'est pas accessible dans le domaine de la fonction `count()`. Toutefois, vous pouvez y accéder si vous lui ajoutez le préfixe `_global`. Par exemple, vous pouvez y accéder si vous ajoutez le préfixe `_global` au compteur, comme dans le code suivant :

```
trace(_global.counter);
```

Vous ne pouvez pas affecter des types de données stricts aux variables créées dans le domaine _global puisque vous devez utiliser le mot-clé var lorsque vous affectez un type de données. Par exemple, vous ne pourriez pas faire :

```
_global.foo:String = "foo";  //erreur de syntaxe
var _global.foo:String = "foo"; //erreur de syntaxe
```

Le Sandbox de Flash Player version 7 et ses versions suivantes applique des restrictions concernant l'accès de variables globales à partir de fichiers SWF chargés à partir de domaines de sécurité distincts. Pour plus d'informations, consultez le Chapitre 16, « Fonctionnement de la sécurité », à la page 675.

Variables de scénario

Les variables de scénario sont disponibles pour tout script dans ce scénario. Pour déclarer des variables de scénario, utilisez l'instruction `var` et initialisez-les dans chaque image du scénario. Cette image et toutes les suivantes peuvent accéder à la variable, comme le montre l'exemple suivant.

Pour utiliser des variables de scénario dans un document :

1. Créez un document Flash et nommez-le **timelinevar.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

   ```
   var myNum:Number = 15; /* initialisé dans l'image 1, donc accessible à
     toutes les images */
   ```
3. Sélectionnez l'image 20 du scénario.
4. Sélectionnez Insertion > Scénario > Image-clé vide.
5. La nouvelle image-clé sélectionnée, tapez le code ActionScript suivant dans le panneau Actions :

   ```
   trace(myNum);
   ```
6. Choisissez Contrôle > Tester l'animation pour tester le nouveau document.

   La valeur 15 apparaît dans le panneau de sortie au bout d'une seconde environ. Les documents Flash fonctionnant en boucle par défaut, la valeur 15 apparaît continuellement dans le panneau de sortie chaque fois que la tête de lecture atteint l'image 20 dans le scénario. Pour interrompre la boucle, ajoutez `stop();` après l'instruction `trace()`.

Vous devez déclarer toute variable de scénario avant d'y accéder dans un script. Par exemple, si vous placez le code `var myNum:Number = 15;` dans l'image 20, aucun des scripts joints aux images précédentes ne peut accéder à `myNum` et sont indéfinis au lieu de contenir la valeur `15`.

Variables locales

Lorsque vous utilisez l'instruction `var` dans un bloc de fonction, vous déclarez des *variables locales*. Lorsque vous déclarez une variable locale dans un bloc de fonction (également appelé *définition de fonction*), elle est définie dans le domaine de ce bloc et expire à la fin du bloc. La variable locale n'existe donc que dans cette fonction.

Par exemple, si vous déclarez une variable nommée `myStr` dans une fonction appelée `localScope`, cette variable ne sera pas disponible en dehors de cette fonction.

```
function localScope():Void {
  var myStr:String = "local";
}
localScope();
trace(myStr); // Non défini, car myStr n'est pas définie globalement
```

Si le nom utilisé pour votre variable locale est déjà déclaré en tant que variable de scénario, la définition locale prévaut sur celle du scénario pendant que la variable locale est dans le domaine. La variable de scénario persiste hors de la fonction. Par exemple, le code suivant crée une variable de scénario de type chaîne appelée `str1`, puis une variable locale du même nom dans la fonction `scopeTest()`. L'instruction `trace` placée dans la fonction génère la définition locale de la variable, tandis que celle qui est placée hors de la fonction génère la définition de scénario de la variable.

```
var str1:String = "Timeline";
function scopeTest():Void {
  var str1:String = "Local";
  trace(str1); // Locale
}
scopeTest();
trace(str1); // Scénario
```

Dans l'exemple suivant, vous constatez que certaines variables n'existent que pour les besoins d'une fonction spécifique et qu'elles peuvent générer des erreurs si vous y faites référence hors du domaine de cette fonction.

Utilisation des variables locales dans une application

1. Créez un nouveau document Flash.
2. Ouvrez le panneau Actions (Fenêtre > Actions) et ajoutez le code ActionScript suivant dans l'image 1 du scénario :

   ```
   function sayHello(nameStr:String):Void {
     var greetingStr:String = "Hello, " + nameStr;
     trace(greetingStr);
   }
   sayHello("world"); // Hello, world
   trace(nameStr); // non défini
   trace(greetingStr); // non défini
   ```

3. Choisissez Contrôle > Tester l'animation pour tester le document.

   Flash affiche la chaîne « Hello, world » dans le panneau de sortie et `undefined` pour les valeurs des variables `nameStr` et `greetingStr` car elles ne sont plus disponibles dans ce domaine. Vous pouvez faire référence à `nameStr` et `greetingStr` uniquement lors de l'exécution de la fonction `sayHello`. Dès que cette fonction se termine, ces variables cessent d'exister.

Les variables `i` et `j` sont souvent utilisées comme compteurs de boucles. Dans l'exemple suivant, vous utilisez `i` comme variable locale qui existe uniquement dans la fonction `initArray()` :

```
var myArr:Array = new Array();
function initArray(arrayLength:Number):Void {
  var i:Number;
  for(i = 0; i < arrayLength; i++) {
    myArr[i] = i + 1;
  }
}
trace(myArr); // <vide>
initArray(3);
trace(myArr); // 1,2,3
trace(i); // non défini
```

REMARQUE

Il n'est pas rare de rencontrer la syntaxe suivante pour une boucle `for` : `for (var i:Number = 0; i < arrayLength; i++) {...}`.

Cet exemple affiche la valeur `undefined` dans l'environnement de test Flash car la variable `i` n'est pas définie dans le scénario principal. Elle existe uniquement dans la fonction `initArray()`.

Vous pouvez également utiliser des variables locales pour empêcher les conflits de noms, qui peuvent donner lieu à des résultats inattendus dans votre application. Par exemple, si vous utilisez `age` comme variable locale, elle vous permet de stocker l'âge d'une personne dans un contexte et celui d'un enfant de la personne dans un autre contexte. Dans ce cas, aucun conflit ne se produit car ces variables sont utilisées dans des domaines distincts.

Il est toujours judicieux d'utiliser des variables locales dans le corps d'une fonction pour que celle-ci puisse agir en tant que partie de code indépendante. Vous ne pouvez modifier une variable locale que dans son bloc de code. Si une expression d'une fonction utilise une variable globale, du code ou des événements extérieurs peuvent modifier sa valeur, ce qui modifierait la fonction.

Vous pouvez affecter un type de données à une variable locale lorsque vous la déclarez, ce qui vous évite d'affecter un type de données erroné à une variable existante. Pour plus d'informations, voir « Affectation des types de données et typage strict », à la page 45.

Chargement des variables

Dans les sections suivantes, vous allez charger des variables depuis le serveur de différentes manières ou dans un document depuis une chaîne d'URL ou FlashVars (qui permet de transférer des variables dans Flash) dans votre code HTML. Ces pratiques présentent les méthodes différentes d'utilisation des variables hors d'un fichier SWF.

Pour plus d'informations sur le chargement des variables (telles que des paires nom/valeur), consultez le Chapitre 15, « Utilisation de données externes », à la page 631.

Vous pouvez utiliser des variables de différentes manières dans un fichier SWF, selon vos besoins. Pour plus d'informations, voir les sections suivantes :

Utilisation de variables depuis l'URL

Lorsque vous développez une application ou un exemple simple dans Flash, vous pourriez souhaiter faire passer des valeurs entre une page HTML et votre document Flash. Les valeurs transmises portent parfois le nom de *chaîne de requête* ou *variables de code URL*. Les variables d'URL s'avèrent très pratiques pour créer un menu dans Flash par exemple. Vous pouvez initialiser le menu pour afficher la navigation appropriée par défaut ou vous pouvez créer une visionneuse dans Flash et définir l'image à afficher par défaut sur le site Web.

Utilisation des variables d'URL dans un document

1. Créez un document Flash et appelez-le **urlvariables.fla**.
2. Sélectionnez Fichier > Enregistrer sous, puis enregistrez le document sur votre ordinateur.
3. Sélectionnez l'image 1 du scénario, puis ajoutez le code suivant dans le panneau Actions :

```
this.createTextField("myTxt", 100, 0, 0, 100, 20);
myTxt.autoSize = "left";
myTxt.text = _level0.myURL;
```

4. Choisissez Contrôle > Tester l'animation pour tester le fichier SWF dans Flash Player.

   Le champ de texte contient la valeur `undefined`. Pour vous assurer que les variables sont bien définies, vous devez vérifier leur présence dans Flash. Pour ce faire, vérifiez leurs valeurs qui doivent être non définies.

5. Pour vérifier si une variable est définie, modifiez le code ActionScript que vous avez ajouté dans le panneau Actions) à l'étape 3 pour qu'il ressemble au code suivant. Ajoutez le code qui apparaît en **gras** :

```
this.createTextField("myTxt", 100, 0, 0, 100, 20);
myTxt.autoSize = "left";
if (_level0.myURL == undefined) {
  myTxt.text = "myURL is not defined";
} else {
  myTxt.text = _level0.myURL;
}
```

Lorsque vous publiez votre document Flash, un document HTML est créé par défaut dans le même répertoire que le fichier SWF. Si un fichier HTML n'a pas été créé, sélectionnez Fichier > Paramètres de publication, et assurez-vous d'avoir sélectionné HTML dans l'onglet Formats. Puis, republiez votre document.

Le code suivant montre les lignes HTML dans le document qui sont responsables de l'imbrication d'un document Flash dans une page HTML. Examinez bien ce code HTML pour comprendre le fonctionnement des variables d'URL dans l'étape suivante (qui consiste à ajouter du code pour les variables d'URL).

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
  codebase="http://fpdownload.adobe.com/pub/shockwave/cabs/flash/
  swflash.cab#version=8,0,0,0" width="550" height="400"
  id="urlvariables" align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="urlvariables.swf" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="urlvariables.swf" quality="high" bgcolor="#ffffff"
  width="550" height="400" name="urlvariables" align="middle"
  allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
  pluginspage="http://www.adobe.com/go/getflashplayer" />
</object>
```

6. Pour faire passer des variables du document HTML généré à votre document Flash, vous pouvez les ajouter à la suite du chemin d'accès et du nom de fichier (urlvariables.swf). Ajoutez le **texte en gras** au fichier HTML généré sur votre ordinateur.

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
  codebase="http://fpdownload.adobe.com/pub/shockwave/cabs/flash/
  swflash.cab#version=8,0,0,0" width="550" height="400"
  id="urlvariables" align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="urlvariables.swf?myURL=http://
  weblogs.macromedia.com" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="urlvariables.swf?myURL=http://weblogs.macromedia.com"
  quality="high" bgcolor="#ffffff" width="550" height="400"
  name="urlvariables" align="middle" allowScriptAccess="sameDomain"
  type="application/x-shockwave-flash" pluginspage="http://
  www.adobe.com/go/getflashplayer" />
</object>
```

7. Si vous souhaitez transmettre plusieurs variables dans Flash, vous devez séparer les paires nom/valeur par un caractère esperluette (&). Récupérez le code suivant à l'étape 6 :

```
?myURL=http://weblogs.macromedia.com
```

Remplacez-le par le texte suivant :

```
?myURL=http://weblogs.macromedia.com&myTitle=adobe+News+Aggregator
```

N'oubliez pas que vous devez modifier de la même manière les deux balises `object` et `embed` pour garantir la cohérence sur tous les navigateurs. Vous remarquerez que les mots sont séparés par des signes +, car les valeurs sont du code URL et que le signe + représente un espace vide.

REMARQUE	Pour connaître la liste des caractères spéciaux de code URL les plus courants, consultez la TechNote Flash, URL Encoding: Reading special characters from a text file.

L'esperluette (`&`) servant de délimiteur pour différentes paires nom/valeur, des résultats inattendus risquent de se produire lorsque les valeurs que vous transmettez en contiennent. Etant donné la nature des paires nom/valeur et de l'analyse, si vous faites passer les valeurs suivantes dans Flash,

```
my.swf?name=PB+&+J&flavor=strawberry+rhubarb
```

Flash créera les variables (et les valeurs) suivantes dans le domaine racine :

```
'name': 'PB ' (note space at end of value)
' J': '' (note space at beginning of variable name and an empty value)
'flavor': 'strawberry rhubarb'
```

Pour éviter cela, vous devez *remplacer* l'esperluette (`&`) dans la paire nom/valeur par son équivalent en code URL (`%26`).

8. Ouvrez le document urlvariables.html et localisez le code suivant :

```
?myURL=http://weblogs.macromedia.com&myTitle=Adobe+News+Aggregator
```

Remplacez-le par le code suivant :

```
?myURL=PB+%26+J&flavor=strawberry+rhubarb
```

9. Enregistrez le code HTML modifié et testez de nouveau votre document Flash.

Vous constatez que Flash crée les paires nom/valeur suivantes.

```
'name': 'PB & J'
'flavor': 'strawberry rhubarb'
```

REMARQUE	Tous les navigateurs prennent en charge les chaînes d'une longueur allant jusqu'à 64 Ko (65 535 octets). Un paramètre FlashVars doit être affecté aux deux balises `object` et `embed` pour que votre code fonctionne dans tous les navigateurs.

Utilisation de FlashVars dans une application

L'emploi de FlashVars pour faire passer des variables dans Flash est similaire à l'emploi d'URL dans le code HTML. Avec FlashVars, au lieu d'être à la suite du nom de fichier, les variables sont transmises dans une balise `param` distincte, ainsi que dans une balise `embed`.

Utilisation de FlashVars dans un document

1. Créez un document Flash et appelez-le **myflashvars.fla**.
2. Sélectionnez Fichier > Paramètres de publication et assurez-vous d'avoir sélectionné HTML, puis cliquez sur OK pour fermer la boîte de dialogue.
3. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
this.createTextField("myTxt", 100, 0, 0, 100, 20);
myTxt.autoSize = "left";
if (_level0.myURL == undefined) {
  myTxt.text = "myURL is not defined";
} else {
  myTxt.text = _level0.myURL;
}
```

REMARQUE	Par défaut, le code HTML publie dans le même emplacement que myflashvars.fla.

4. Sélectionnez Fichier > Publier pour publier les fichiers SWF et HTML.
5. Accédez au répertoire contenant les fichiers publiés (celui où vous avez enregistré le fichier myflashvars.fla dans votre disque dur), puis ouvrez le document HTML (myflashvars.html par défaut) dans un éditeur HTML, tel que Dreamweaver ou Notepad.
6. Ajoutez le code qui apparaît en **gras** ci-dessous pour que votre document HTML ressemble au code suivant :

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
  codebase="http://fpdownload.adobe.com/pub/shockwave/cabs/flash/
  swflash.cab#version=8,0,0,0" width="550" height="400" id="myflashvars"
  align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="myflashvars.swf" />
<param name="FlashVars" value="myURL=http://weblogs.adobe.com/">
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="myflashvars.swf" FlashVars="myURL=http://weblogs.adobe.com/"
  quality="high" bgcolor="#ffffff" width="550" height="400"
  name="myflashvars" align="middle" allowScriptAccess="sameDomain"
  type="application/x-shockwave-flash" pluginspage="http://
  www.adobe.com/go/getflashplayer" />
</object>
```

Ce code fait passer une seule variable appelée `myURL`, qui contient la chaîne `http://weblogs.macromedia.com`. Lors du chargement du fichier SWF, une propriété appelée `myURL` est créée dans le domaine _level0. L'un des avantages de l'emploi de FlashVars ou de l'URL réside dans la disponibilité immédiate des variables dans Flash dès le chargement du fichier SWF. Ainsi, il est inutile d'écrire des fonctions supplémentaires pour vérifier le chargement des variables, ce qui serait obligatoire si vous les chargiez à l'aide de LoadVars ou XML.

7. Enregistrez vos modifications du document HTML, puis fermez-le.
8. Double-cliquez sur le fichier myflashvars.html pour tester l'application.

   Le texte `http://weblogs.macromedia.com`, une variable du fichier HTML, apparaît dans le fichier SWF.

REMARQUE Tous les navigateurs prennent en charge les chaînes d'une longueur allant jusqu'à 64 Ko (65535 octets). Un paramètre FlashVars doit être affecté aux deux balises `object` et `embed` pour que votre code fonctionne dans tous les navigateurs.

Chargement des variables depuis un serveur

Il existe plusieurs méthodes pour charger des variables dans Flash à partir de sources externes (telles que des fichiers texte, des documents XML, etc.). Pour plus d'informations sur le chargement des variables (y compris des paires nom/valeur), consultez le Chapitre 15, « Utilisation de données externes », à la page 631.

Dans Flash, la classe LoadVars permet de charger facilement des variables, comme le montre l'exemple suivant.

Chargement des variables depuis un serveur

1. Créez un document Flash.
2. Sélectionnez l'image 1 du scénario, puis ajoutez le code ActionScript suivant dans le panneau Actions :

```
var my_lv:LoadVars = new LoadVars();
my_lv.onLoad = function(success:Boolean):Void {
  if (success) {
    trace(this.dayNames); // Sunday,Monday,Tuesday,...
  } else {
    trace("Error");
  }
}
my_lv.load("http://www.helpexamples.com/flash/params.txt");
```

   Ce code charge un fichier texte depuis un serveur distant et analyse ses paires nom/valeur.

CONSEIL Téléchargez ou affichez le fichier texte (http://www.helpexamples.com/flash/params.txt) dans un navigateur si vous souhaitez savoir comment les variables sont mises en forme.

3. Choisissez Contrôle > Tester l'animation pour tester le document.

   Si le chargement réussi, l'événement `complete` est appelé et le panneau de sortie affiche la valeur de la variable `dayNames`. Si le chargement du fichier texte échoue, l'argument `success` est défini sur `false` et le panneau de sortie affiche le texte `Error`.

Utilisation des variables dans un projet

Lorsque vous développez des animations ou des applications avec Flash, il est très rare que vous n'ayez besoin d'aucune variable. Par exemple, si vous créez un système d'ouverture de session, vous aurez besoin de variables pour valider le nom d'utilisateur et le mot de passe ou vérifier leur remplissage.

Pour plus d'informations sur le chargement des variables (telles que des paires nom/valeur), consultez le Chapitre 15, « Utilisation de données externes », à la page 631.

Dans l'exemple suivant, vous utilisez des variables pour stocker le chemin d'une image que vous chargez avec la classe Loader, une pour l'occurrence de la classe Loader et deux fonctions sont appelées selon le résultat, satisfaisant ou non, du chargement du fichier.

Utilisation des variables dans un projet

1. Créez un document Flash, puis enregistrez-le sous le nom de **imgloader.fla**.
2. Sélectionnez l'image 1 du scénario, puis ajoutez le code ActionScript suivant dans le panneau Actions :

```
/* Spécifie une image par défaut au cas où aucune valeur n'est transmise
   par FlashVars. */
var imgUrl:String = "http://www.helpexamples.com/flash/images/
   image1.jpg";
if (_level0.imgURL != undefined) {
   // Si une image a été désignée, la valeur par défaut est remplacée.
   imgUrl = _level0.imgURL;
}

this.createEmptyMovieClip("img_mc", 10);
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip):Void {
   target_mc._x = (Stage.width - target_mc._width) / 2;
   target_mc._y = (Stage.height - target_mc._height) / 2;
}
mclListener.onLoadError = function(target_mc:MovieClip):Void {
   target_mc.createTextField("error_txt", 1, 0, 0, 100, 20);
   target_mc.error_txt.autoSize = "left";
   target_mc.error_txt.text = "Error downloading specified image;\n\t" +
   target_mc._url;
}
var myMCL:MovieClipLoader = new MovieClipLoader();
myMCL.addListener(mclListener);
myMCL.loadClip(imgUrl, img_mc);
```

La première ligne du code désigne l'image à charger dynamiquement dans votre document Flash. Ensuite, vous vérifiez si une nouvelle valeur a été spécifiée pour `imgURL` à l'aide de paramètres FlashVars ou de variables de code URL. Dans l'affirmative, l'URL de l'image par défaut est remplacée par la nouvelle valeur. Pour plus d'informations sur l'utilisation des variables d'URL, consultez la section « Utilisation de variables depuis l'URL », à la page 67. Pour plus d'informations sur FlashVars, consultez la section « Utilisation de FlashVars dans une application », à la page 70.

Les deux lignes de code suivantes définissent l'occurrence de MovieClip et un objet écouteur pour la future occurrence de MovieClipLoader. L'objet écouteur du MovieClipLoader définit deux gestionnaires d'événement, `onLoadInit` et `onLoadError`. Ces gestionnaires sont invoqués lorsque le chargement et l'initialisation de l'image sur la scène ont réussi, ou quand le chargement a échoué. Ensuite, vous créez une occurrence de MovieClipLoader et vous utilisez la méthode `addListener()` pour ajouter l'objet écouteur défini préalablement au MovieClipLoader. Pour finir, l'image est téléchargée et déclenchée par votre appel de la méthode `MovieClipLoader.loadClip()`, qui désigne le fichier de l'image à charger et le clip qui doit la recevoir.

3. Choisissez Contrôle > Tester l'animation pour tester le document.

   Comme vous testez le document Flash dans l'outil de programmation, aucune valeur de `imgUrl` n'est transmise par FlashVars ni avec l'URL. Par conséquent, l'image par défaut s'affiche.

4. Enregistrez le document Flash, puis sélectionnez Fichier > Publier pour publier le fichier sous forme de document SWF et HTML.

   REMARQUE Assurez-vous que les deux options Flash et HTML sont bien sélectionnées dans la boîte de dialogue Paramètres de publication. Choisissez Fichier › Paramètres de publication et cliquez sur l'onglet Formats. Sélectionnez ensuite les deux options.

5. Si vous testez votre document dans l'outil de Flash (Contrôle > Tester l'animation) ou dans un navigateur local (Fichier > Aperçu avant publication > HTML), vous constaterez que l'image se centre elle-même verticalement et horizontalement sur la scène.

6. Modifiez le document HTML généré dans un éditeur (tel que Dreamweaver ou Notepad), puis modifiez le code HTML par défaut pour qu'il corresponde au texte suivant :

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
  codebase="http://fpdownload.adobe.com/pub/shockwave/cabs/flash/
  swflash.cab#version=8,0,0,0" width="550" height="400" id="imgloader"
  align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="imgloader.swf" />
<param name="FlashVars" value="imgURL=http://www.helpexamples.com/flash/
  images/image2.jpg">
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="imgloader.swf" quality="high" FlashVars="imgURL=http://
  www.helpexamples.com/flash/images/image2.jpg" bgcolor="#ffffff"
  width="550" height="400" name="imgloader" align="middle"
  allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
  pluginspage="http://www.adobe.com/go/getflashplayer" />
</object>
```

7. Testez le document HTML pour examiner les changements. L'image spécifiée dans le code HTML apparaît dans le fichier SWF.

   Pour modifier cet exemple afin d'utiliser vos propres images, modifiez la valeur FlashVars (chaîne entre guillemets).

Organisation des données dans des objets

Vous êtes probablement habitué(e) à manipuler les objets placés sur la scène. Supposons par exemple que vous avez un objet MovieClip sur la scène, qui contient lui-même d'autres clips. Les champs de texte, les clips et les boutons sont souvent considérés comme des objets dès qu'ils sont déposés sur la scène.

Dans ActionScript, les objets sont des collections de propriétés et de méthodes. Chaque objet possède son propre nom et est une occurrence d'une classe donnée. Les objets intégrés proviennent de classes qui sont prédéfinies dans ActionScript. Par exemple, la classe intégrée Date fournit des informations provenant de l'horloge système de l'ordinateur de l'utilisateur. Vous pouvez utiliser la classe LoadVars pour charger des variables dans votre fichier SWF.

Vous pouvez également créer des objets et des classes avec du code ActionScript. Vous pouvez créer un objet destiné à stocker une collection de données, telles que le nom, l'adresse et le numéro de téléphone d'une personne. Vous pouvez créer un objet destiné à stocker des informations sur les couleurs d'une image. L'organisation des données dans des objets permet de tenir vos documents Flash bien ordonnés. Pour des informations d'ordre général sur la création d'une classe personnalisée destinée à stocker une collection de méthodes et de propriétés, consultez la section « Ecriture de fichiers de classe personnalisée », à la page 208. Pour des informations détaillées sur les classes intégrées et personnalisées, consultez le Chapitre 6, « Classes », à la page 197.

Il existe plusieurs manières de créer un objet dans ActionScript. L'exemple suivant crée des objets simples de deux manières différentes, puis passe en boucle sur leur contenu.

Création d'objets simples dans Flash

1. Créez un document Flash, puis enregistrez-le sous le nom **simpleObjects.fla**.
2. Sélectionnez l'image 1 du scénario et, dans le panneau Actions, tapez le code ActionScript suivant :

```
// Première méthode
var firstObj:Object = new Object();
firstObj.firstVar = "hello world";
firstObj.secondVar = 28;
firstObj.thirdVar = new Date(1980, 0, 1); // 1er janvier 1980
```

   Ce code, qui constitue une méthode pour créer un objet simple, crée une nouvelle occurrence d'objet et définit quelques propriétés dans cet objet.

3. Maintenant, tapez le code ActionScript suivant après celui ajouté à l'étape 2.

```
// Deuxième méthode
var secondObj:Object = {firstVar:"hello world", secondVar:28,
  thirdVar:new Date(1980, 0, 1)};
```

   Voici une autre manière de créer un objet. Les deux objets sont semblables. Le code ci-dessus crée un nouvel objet et initialise certaines propriétés sous forme courte.

4. Pour passer en boucle sur chacun des objets précédents et afficher leur contenu, ajoutez le code ActionScript suivant sur l'image 1 du scénario (à la suite du code déjà saisi) :

```
var i:String;
for (i in secondObj) {
  trace(i + ": " + secondObj[i]);
}
```

5. Choisissez Contrôle > Tester l'animation pour voir le texte suivant dans le panneau de sortie :

```
firstVar: hello world
secondVar: 28
thirdVar: Tue Jan 1 00:00:00 GMT-0800 1980
```

REMARQUE	Il se peut que les variables n'apparaissent pas nécessairement dans cet ordre dans le panneau de sortie. Vous ne pouvez pas garantir l'ordre lorsque vous utilisez une boucle for..in ; le lecteur renvoie le contenu d'un objet dans un ordre imprévisible.

Vous pouvez également utiliser des tableaux pour créer des objets. Au lieu d'avoir une série de variables telles que `firstname1`, `firstname2` et `firstname3` pour représenter une collection de variables, vous pouvez fabriquer un tableau contenant les mêmes données. Cette technique est présentée ci-après.

Utilisation d'un tableau pour créer un objet

1. Créez un document Flash, puis enregistrez-le sous le nom **arrayObject.fla**.
2. Sélectionnez l'image 1 du scénario et, dans le panneau Actions, tapez le code ActionScript suivant :

```
var usersArr:Array = new Array();
usersArr.push({firstname:"George"});
usersArr.push({firstname:"John"});
usersArr.push({firstname:"Thomas"});
```

   L'organisation des variables dans des tableaux et des objets présente l'avantage de faciliter le passage en boucle sur les variables et l'affichage de leurs valeurs, comme le montre l'étape suivante.

3. Tapez le code ActionScript suivant après celui ajouté à l'étape 2.

```
var i:Number;
for (i = 0; i < usersArr.length; i++) {
  trace(usersArr[i].firstname); // George, John, Thomas
}
```

4. Choisissez Contrôle > Tester l'animation pour voir le texte suivant dans le panneau de sortie :

```
George
John
Thomas
```

L'exemple suivant présente une autre méthode de passage en boucle sur les objets. Ici, un objet est créé et survolé répétitivement à l'aide d'une boucle `for..in` et chaque propriété apparaît dans le panneau de sortie :

```
var myObj:Object = {var1:"One", var2:"Two", var3:18, var4:1987};
var i:String;
for (i in myObj) {
  trace(i + ": " + myObj[i]);
}
// Résultats suivants :
/*
    var1: One
    var2: Two
    var3: 18
    var4: 1987
*/
```

Pour plus d'informations sur les boucles, consultez le Chapitre 4, « Utilisation des boucles for », à la page 126. Pour plus d'informations sur les boucles for..in, consultez la section « Utilisation des boucles for..in », à la page 127. Pour plus d'informations sur les objets, consultez la section Chapitre 6, « Classes », à la page 197.

Attribution

ActionScript 2.0 vous permet d'attribuer un type de données à un autre. Cette opération signifie que vous convertissez la valeur que l'objet ou la variable contient en un type différent.

Les résultats de cette attribution dépendent des types de données impliqués. Pour changer le type de données d'un objet, vous mettez le nom de l'objet entre parenthèses (`()`) et le faites précéder du nom du nouveau type. Par exemple, le code suivant prend une valeur booléenne et la transforme en entier numérique.

```
var myBoolean:Boolean = true;
var myNumber:Number = Number(myBoolean);
```

Pour plus d'informations sur l'attribution, consultez les sections suivantes :

- « Attribution des objets », à la page 78

Attribution des objets

La syntaxe de l'attribution est `type(élément)` : le compilateur doit se comporter comme si le type de données de l'élément était `type`. L'attribution est avant tout un appel de fonction qui renvoie `null` si l'attribution échoue lors de l'exécution (dans les fichiers publiés pour Flash Player version 7 ou plus récente). L'exécution des fichiers publiés pour Flash Player 6 n'est pas prise en charge en cas d'échec de l'attribution. Si l'attribution réussit, l'appel de fonction renvoie l'objet original. Cependant, le compilateur ne peut pas déterminer si une attribution échoue lors de l'exécution et ne génère pas d'erreurs de compilation dans ces cas.

Le code suivant illustre cette situation :

```
// Les classes Cat et Dog sont des sous-classes de la classe Animal
function bark(myAnimal:Animal) {
  var foo:Dog = Dog(myAnimal);
  foo.bark();
}
var curAnimal:Animal = new Dog();
bark(curAnimal); // Fonctionne
curAnimal = new Cat();
bark(curAnimal); // Ne fonctionne pas
```

Dans cet exemple, vous avez indiqué au compilateur que `foo` est un objet Dog, et le compilateur suppose donc que `foo.bark()` est une instruction autorisée. Toutefois, le compilateur ne sait pas que l'attribution échouera (c'est-à-dire que vous avez tenté d'attribuer un objet Cat au type Animal), et aucune erreur de compilation ne se produit. Toutefois, si vous incorporez une vérification dans votre script de manière à vous assurer que l'attribution réussit, vous pouvez trouver des erreurs d'attribution à l'exécution, comme le montre l'exemple suivant.

```
function bark(myAnimal:Animal) {
  var foo:Dog = Dog(myAnimal);
  if (foo) {
    foo.bark();
  }
}
```

Vous pouvez attribuer une expression à une interface. Si l'expression est un objet qui implémente l'interface ou si elle possède une classe de base qui implémente l'interface, l'attribution réussit. Sinon, l'attribution échoue.

REMARQUE	L'attribution de valeurs Null ou non définies renvoie `undefined`.

Vous pouvez remplacer les types de données primitifs qui disposent d'une fonction de conversion globale correspondante par un opérateur d'attribution du même nom. C'est pourquoi les fonctions de conversion globales sont prioritaires par rapport aux opérateurs d'attribution. Par exemple, vous ne pouvez pas attribuer Array dans la mesure où la fonction de conversion `Array()` est prioritaire par rapport à l'opérateur d'attribution.

Cet exemple définit deux variables de chaîne (`firstNum` et `secondNum`), qui s'ajoutent l'une à l'autre. Résultat initial : les nombres sont concaténés au lieu d'être ajoutés car ils sont de type String. La deuxième instruction trace convertit les deux nombres en type de données Number avant d'exécuter l'addition qui donne le résultat correct. La conversion des données est importante lorsque vous manipulez des données chargées à l'aide de XML ou FlashVars, comme le montre l'exemple suivant :

```
var firstNum:String = "17";
var secondNum:String = "29";
trace(firstNum + secondNum); // 1729
trace(Number(firstNum) + Number(secondNum)); // 46
```

Pour plus d'informations sur les fonctions de conversion des données, consultez l'entrée de chaque fonction de conversion dans le *Guide de référence du langage ActionScript 2.0* : `Fonction Array`, `fonction Boolean`, `fonction Number`, `fonction Object` et `fonction String`.

CHAPITRE 4

Eléments fondamentaux du langage et de la syntaxe

4

L'apprentissage de la syntaxe et des instructions d'ActionScript revient à apprendre à associer des mots pour construire des phrases, à rassembler ensuite au sein de paragraphes. Le langage ActionScript atteint ce niveau de simplicité. Par exemple, en français, une phrase se termine par un point, tandis que les instructions de code ActionScript se terminent par un point-virgule. En langage ActionScript, vous pouvez taper une instruction `stop()` pour interrompre la lecture en boucle d'une occurrence de clip ou d'un fichier SWF. Vous pouvez également écrire des milliers de lignes de code pour mettre au point une application bancaire interactive. ActionScript peut donc accomplir des tâches très simples ou très complexes.

Dans le Chapitre 3, « Données et types de données », vous allez découvrir comment ActionScript manipule les données et comment les mettre en forme dans votre code. Ce chapitre décrit la rédaction d'instructions dans ActionScript avec la *syntaxe* appropriée. Il contient un grand nombre de petits blocs de code et quelques exemples qui présentent les concepts de base de ce langage. Les chapitres suivants contiennent des exemples de code plus longs et de plus en plus complexes qui combinent et simplifient à la fois les bases présentées dans ce chapitre.

Les règles générales décrites dans cette section s'appliquent à l'ensemble du code ActionScript. La plupart des termes ActionScript font également l'objet de règles individuelles. Pour en savoir plus sur les règles qui s'appliquent à un terme particulier, reportez-vous à l'entrée correspondante du *Guide de référence du langage ActionScript 2.0*.

Les nouveaux utilisateurs peuvent éprouver des difficultés à créer des programmes ActionScript de façon élégante. Pour plus de détails sur l'application des règles décrites dans cette section, consultez le Chapitre 17, « Recommandations et conventions de programmation pour ActionScript 2.0 », à la page 715.

REMARQUE

Dans ce chapitre, vous allez ajouter du code ActionScript directement dans une image du scénario. Dans les chapitres suivants, vous utiliserez des classes pour isoler votre code ActionScript du fichier FLA.

Pour plus d'informations sur l'utilisation des concepts de base du langage et de la syntaxe ActionScript, consultez les sections suivantes :

Présentation de la syntaxe, des instructions et des expressions

Le langage ActionScript est composé de classes intégrées. Pour formuler vos instructions de sorte que le code puisse être compilé et exécuté correctement dans Flash, il est nécessaire d'utiliser la *syntaxe* ActionScript appropriée. Ici, le terme syntaxe fait référence à la grammaire et à l'orthographe du langage de programmation. Le compilateur ne pouvant pas interpréter la syntaxe incorrecte, des erreurs ou des avertissements apparaissent dans le panneau de sortie lorsque le document est testé dans l'environnement de test. La syntaxe est donc un ensemble de règles et de consignes qui vous aide à rédiger correctement votre code ActionScript.

Une *instruction* est une action spécifique que vous demandez au fichier FLA d'exécuter. Par exemple, vous pouvez utiliser une instruction conditionnelle pour déterminer si quelque chose est vrai ou existe. Vous pouvez ensuite exécuter les actions spécifiées, telles que des fonctions ou des expressions, selon si la condition est vraie ou non. L'instruction `if` est une instruction conditionnelle qui évalue une condition afin de déterminer la prochaine action du code.

```
// instruction if
if (condition) {
  // instructions ;
}
```

Pour plus d'informations sur les instructions, consultez la section « Présentation des instructions », à la page 109.

Une *expression* diffèr d'une instruction et désigne toute combinaison valable de symboles ActionScript représentant une valeur. Les expressions ont des *valeurs* alors que les valeurs et les propriétés ont des *types*. Une expression peut être composée d'opérateurs et d'opérandes, de valeurs, de fonctions et de procédures, et suit les règles de priorité et d'association d'ActionScript. En général, Flash Player interprète l'expression et renvoie une valeur que vous pouvez utiliser dans votre application.

Par exemple, le code suivant est une expression :

```
x + 2
```

Dans l'expression précédente, `x` et `2` sont des opérandes et + est un opérateur. Pour plus d'informations sur les opérateurs et les opérandes, consultez la section « Présentation des opérateurs », à la page 145. Pour plus d'informations sur les objets et les propriétés, consultez la section « Type de données Object », à la page 43.

La mise en forme de votre code ActionScript conditionne également la facilité de sa maintenance. Par exemple, il est extrêmement difficile de comprendre la logique d'un fichier FLA qui ne comporte ni indentation ni commentaire, ou dont le formatage et les conventions d'appellation sont incohérents. Lorsque les blocs ActionScript (tels que les boucles et les instructions `if`) sont placés en retrait, le code est plus facile à lire et à déboguer en cas d'anomalie. Pour plus d'informations sur la mise en forme du code ActionScript, consultez la section « Mise en forme de la syntaxe ActionScript », à la page 750. Ces sections contiennent également des exemples de mise en forme appropriée du code ActionScript.

Pour plus d'informations sur les concepts de base du langage et de la syntaxe, consultez les sections suivantes :

- « Différences entre ActionScript et JavaScript »
- « Respect de la casse »

Différences entre ActionScript et JavaScript

ActionScript est analogue au langage de programmation JavaScript. Pour utiliser le code ActionScript, la connaissance de JavaScript n'est pas indispensable, mais elle en facilite considérablement l'apprentissage.

Cet ouvrage n'a pas pour but d'enseigner la programmation générale. Il existe de nombreuses sources qui fournissent des informations complémentaires sur les concepts de programmation généraux et sur le langage JavaScript.

- La spécification ECMAScript (ECMA-262) édition 3 est issue de JavaScript et sert de norme internationale pour le langage JavaScript. ActionScript est basé sur cette spécification. Pour plus d'informations, consultez le site www.ecma-international.org/publications/standards/Ecma-262.htm.
- Le site Java Technology propose des didacticiels sur la programmation orientée objet (http://java.sun.com/docs/books/tutorial/java/index.html) qui portent essentiellement sur le langage Java, mais permettent de comprendre les concepts qui s'appliquent également à ActionScript.

Les principales différences qui existent entre ActionScript et JavaScript sont les suivantes :

- ActionScript ne prend pas en charge les objets spécifiques aux navigateurs que sont les documents, fenêtres et ancres, par exemple.
- ActionScript ne prend pas entièrement en charge tous les objets JavaScript intégrés.
- ActionScript ne prend pas en charge certaines constructions syntaxiques JavaScript, telles que les étiquettes d'instructions.
- Dans ActionScript, la fonction `eval()` ne peut effectuer que des références aux variables.
- ActionScript 2.0 prend en charge plusieurs fonctionnalités qui ne figurent pas dans la spécification ECMA-262, telles que les classes et le typage fort. Un grand nombre de ces fonctionnalités sont modélisées d'après la spécification ECMAScript (ECMA-262) édition 3 (voir www.ecma-international.org/publications/standards/Ecma-262.htm).
- ActionScript ne prend pas en charge les expressions régulières ayant recours à l'objet RegExp, contrairement à Adobe Central, qui prend en charge cet objet. Pour plus d'informations sur Adobe Central, consultez le site www.adobe.com/products/central.

Respect de la casse

Lorsque vous rédigez du code ActionScript pour Flash Player 7 et ses versions ultérieures, votre code respecte la casse. Cela signifie que les variables dont la casse diffère sont considérées comme différentes. Le code ActionScript suivant illustre cette particularité :

```
// Combinaison de majuscules/minuscules
var firstName:String = "Jimmy";
// Tout en minuscules
trace(firstname); // non défini
```

Vous pouvez également écrire :

```
// Dans un fichier destiné à Flash Player 8
// et avec ActionScript 1.0 ou 2.0
//
// Définit les propriétés de deux objets différents
cat.hilite = true;
CAT.hilite = true;

// Crée trois variables différentes
var myVar:Number = 10;
var myvar:Number = 10;
var mYvAr:Number = 10;
```

REMARQUE

Il n'est pas conseillé de différencier les variables, ou tout autre identifiant, uniquement par le changement de la casse. Pour plus d'informations sur l'appellation des variables, consultez le Chapitre 17, « Recommandations et conventions de programmation pour ActionScript 2.0 », à la page 715.

Lorsque vous publiez pour les anciennes versions de Flash Player (version 6 et antérieures), le logiciel recherche la chaîne `Jimmy` dans le panneau de sortie. Flash Player 7 et les versions ultérieures étant sensibles à la casse, `firstName` et `firstname` sont deux variables distinctes (avec ActionScript 1.0 ou 2.0). Ce concept est essentiel. Si vous avez créé des fichiers FLA pour Flash Player 6 ou une version antérieure en utilisant indifféremment la casse dans vos variables, des problèmes risquent de survenir lors de la conversion du fichier ou de l'application pour une version plus récente de Flash Player.

Il est donc fortement conseillé d'adopter des conventions cohérentes en matière d'utilisation des majuscules/minuscules, comme celles employées dans ce manuel. Vos variables, vos classes et les noms de vos fonctions seront ainsi plus simples à distinguer. Ne vous contentez pas de changer la casse pour différencier deux identifiants. Modifiez plutôt le nom de l'occurrence, de la variable ou de la classe, pas seulement sa casse. Pour plus d'informations sur les conventions de codage, consultez le Chapitre 17, « Recommandations et conventions de programmation pour ActionScript 2.0 », à la page 715.

Le respect de la casse peut avoir un impact considérable lorsque vous travaillez avec un service Web qui utilise ses propres règles pour l'appellation des variables et la casse des variables renvoyées par le serveur au fichier SWF. Par exemple, si vous utilisez un service Web ColdFusion, les noms de propriété provenant d'une structure ou d'un objet peuvent rester en majuscules, comme `FIRSTNAME`. Vous devez conserver la même casse dans Flash pour ne pas provoquer de résultats imprévus.

REMARQUE

Le respect de la casse affecte également les variables externes que vous chargez dans un fichier SWF, par exemple celles qui sont chargées avec `LoadVars.load()`.

La différentiation des majuscules et des minuscules est implémentée pour les scripts externes, tels que les fichiers de classe ActionScript 2.0, les scripts importés avec la commande `#include` et les scripts d'un fichier FLA. Si vous subissez des erreurs lors de l'exécution et exportez vos données vers plusieurs versions de Flash Player, vous devez vérifier les scripts externes et les scripts des fichiers FLA pour vous assurer qu'ils appliquent les mêmes majuscules.

Cette différentiation est implémentée au niveau de chaque fichier SWF. Lorsqu'une application Flash Player 8 stricte (qui respecte la casse) appelle un fichier SWF Flash Player 6 non strict, le code ActionScript exécuté dans ce dernier ne respecte pas la casse. Par exemple, si vous utilisez `loadMovie()` pour charger un fichier SWF Flash Player 6 dans un fichier SWF Flash Player 8, celui de la version 6 ne tient pas compte des majuscules, contrairement à celui de la version 8.

Lorsque la coloration de la syntaxe est activée, les éléments du langage dont la casse est correcte apparaissent en bleu par défaut. Pour plus d'informations, voir « Présentation des mots réservés », à la page 109.

Syntaxe à points et chemins cibles

Dans ActionScript, un opérateur point (.) (*syntaxe à point*) est utilisé pour accéder aux propriétés ou méthodes associées à un objet ou à une occurrence de la scène. Il est également utilisé pour identifier le chemin cible d'une occurrence (telle qu'un clip), d'une variable, d'une fonction ou d'un objet.

Une expression en syntaxe à point commence par le nom de l'objet ou du clip suivi d'un point et se termine par l'élément que vous souhaitez spécifier. La rédaction des expressions en syntaxe à point est décrite dans les sections suivantes.

Pour contrôler un clip, un fichier SWF chargé ou un bouton, vous devez spécifier un *chemin cible*. Les chemins cibles sont des adresses hiérarchiques de noms d'occurrences de clips, de variables et d'objets dans un fichier SWF. Pour spécifier le chemin cible d'un clip ou d'un bouton, vous devez lui affecter un nom d'occurrence. Pour nommer une occurrence de clip, sélectionnez-la et saisissez son nom dans l'inspecteur Propriétés. Il est également possible de nommer l'occurrence dans le code si vous la créez avec ActionScript. Vous pouvez utiliser le chemin cible pour affecter une action à un clip ou pour obtenir ou définir la valeur d'une variable ou propriété.

Pour plus d'informations sur la nomination d'une occurrence et l'utilisation de la syntaxe à point pour cibler une occurrence, consultez les sections suivantes :

- « Utilisation de la syntaxe à point pour cibler une occurrence », à la page 86.
- « Domaine et ciblage », à la page 91
- « Utilisation du bouton Chemin cible », à la page 93
- « Syntaxe à barre oblique », à la page 93

Pour plus d'informations sur les objets et les propriétés, consultez la section « Type de données Object », à la page 43.

Utilisation de la syntaxe à point pour cibler une occurrence

Pour rédiger un code ActionScript qui contrôle une occurrence, telle qu'un clip, ou manipule les éléments d'un fichier SWF chargé, vous devez indiquer son nom et son adresse dans le code, c'est-à-dire son *chemin cible*. Pour cibler (ou faire référence à) des objets dans un fichier SWF, vous utilisez la syntaxe à point (également appelée *notation avec point*). Par exemple, vous devez cibler une occurrence de clip ou de bouton avant de pouvoir lui appliquer une action. La syntaxe à point vous aide à créer le chemin vers l'occurrence cible. Ce chemin vers l'occurrence ciblée est parfois appelé chemin cible.

La hiérarchie des fichiers FLA est particulière. Vous pouvez créer des occurrences sur la scène ou utiliser ActionScript. Vous pouvez même créer des occurrences à l'intérieur d'autres occurrences ou en imbriquer au sein de plusieurs autres occurrences. Tant que vous la nommez, vous pouvez manipuler n'importe quelle occurrence.

Les occurrences sont nommées par l'attribution d'un *nom d'occurrence*, que vous pouvez spécifier de deux manières différentes (présentées ci-dessous) :

- Manuellement en sélectionnant l'occurrence et en saisissant son nom dans l'inspecteur Propriétés (lorsque l'occurrence est sur la scène).
- Dynamiquement avec ActionScript. Dans ce cas, vous créez l'occurrence avec ActionScript et lui affectez un nom dès sa création.

Pour affecter un nom d'occurrence à l'occurrence dans l'inspecteur de propriétés, tapez un nom dans la zone de texte Nom de l'occurrence.

Vous pouvez également attribuer un nom d'occurrence à un objet créé à l'aide d'ActionScript. Comme le montre le code suivant, cette opération peut être très simple :

```
this.createEmptyMovieClip("pic_mc", this.getNextHighestDepth());
pic_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
```

Ce code crée un nouveau clip et lui affecte le nom d'occurrence pic_mc. Vous pouvez ensuite manipuler l'occurrence `pic_mc` à l'aide du code, par exemple en y chargeant une image, comme l'illustre le code précédent.

Pour plus d'informations sur l'utilisation des domaines, consultez les sections « Domaine et ciblage », à la page 91 et « Variables et domaine », à la page 62.

Ciblage d'une occurrence

Pour qu'une occurrence fonctionne dans votre fichier SWF, vous devez la cibler, puis lui indiquer quoi faire, par exemple lui affecter une action à exécuter ou modifier ses propriétés. En général, il est nécessaire de définir l'emplacement de cette occurrence dans le fichier SWF (par exemple le scénario dans lequel elle se trouve ou l'occurrence dans laquelle elle est imbriquée) par la création d'un chemin cible. N'oubliez pas que vous avez fourni la plupart des occurrences via leurs noms d'occurrence dans votre fichier FLA, puis ajouté du code dans le fichier FLA qui utilise ces noms. Dans ce cas, vous ciblez cette occurrence particulière, puis lui indiquez une action à exécuter (par exemple déplacer la tête de lecture ou ouvrir une page Web). Pour plus d'informations sur les objets et les propriétés, consultez la section « Type de données Object », à la page 43.

Pour cibler une occurrence :

1. Choisissez Fichier > Nouveau, puis sélectionnez un document Flash.
2. Choisissez Fichier > Enregistrer sous et nommez le fichier **target.fla**.
3. A l'aide de l'outil Ovale, tracez une forme sur la scène. Dessinez un ovale de taille et de couleur quelconques.
4. Utilisez l'outil Sélection pour sélectionner l'ovale sur la scène.

CONSEIL	N'oubliez pas, au besoin, de sélectionner le trait et le remplissage.

5. Choisissez Modifier > Convertir en symbole, sélectionnez l'option Clip, puis cliquez sur OK pour créer le symbole.
6. Sélectionnez le clip sur la scène et donnez-lui le nom d'occurrence **myClip** dans l'inspecteur Propriétés.
7. Insérez un nouveau calque et renommez-le **actions**.
8. Ajoutez le code ActionScript suivant dans l'image 1 du calque actions :

   ```
   myClip._xscale = 50;
   ```

   Cette ligne de code cible l'occurrence `myClip` sur la scène. Le code ActionScript réduit de moitié la largeur d'origine de l'occurrence. Le code étant sur le même scénario que le symbole de clip, il suffit de cibler l'occurrence par son nom. Si cette occurrence se trouvait sur un autre scénario ou était imbriquée dans une autre occurrence, le chemin cible devrait être modifié en conséquence.

Ciblage d'une occurrence imbriquée

Il est également possible de cibler des occurrences imbriquées dans d'autres occurrences. Supposons par exemple que vous souhaitez placer une seconde occurrence de clip dans l'occurrence myClip de l'exercice présenté à la section « Ciblage d'une occurrence », à la page 87. Vous pouvez également cibler cette occurrence imbriquée à l'aide d'ActionScript. Avant de commencer l'exercice suivant, vous devez terminer l'exercice de la section « Ciblage d'une occurrence », à la page 87, puis exécuter cette procédure pour cibler une occurrence imbriquée.

Pour cibler une occurrence imbriquée :

1. Ouvrez le fichier target.fla à partir de la procédure sur le ciblage d'une occurrence et renommez-le **target2.fla**.
2. Double-cliquez sur l'occurrence myClip sur la scène.
3. Sélectionnez l'outil Ovale et tracez un autre ovale dans l'occurrence myClip.
4. Sélectionnez la nouvelle forme, puis choisissez Modifier > Convertir en symbole.
5. Sélectionnez l'option Clip et cliquez sur OK.
6. Sélectionnez la nouvelle occurrence et saisissez **myOtherClip** dans le champ Nom de l'occurrence de l'inspecteur Propriétés.
7. Dans la barre d'édition, cliquez sur Scène 1 pour revenir au scénario principal.
8. Ajoutez le code ActionScript suivant dans l'image 1 du calque actions :

   ```
   myClip.myOtherClip._xscale = 50;
   ```

   Ce code ActionScript réduit de 50 % la largeur de l'occurrence myOtherClip. Le fichier target.fla ayant modifié la propriété `_xscale` des occurrences myClip, et myOtherClip étant un symbole imbriqué, vous remarquerez que la largeur de myOtherClip est maintenant de 25 % de la largeur d'origine.

Si vous travaillez avec des clips imbriqués possédant leurs propres scénarios, vous pouvez manipuler la tête de lecture dans le scénario d'une occurrence imbriquée à l'aide d'un code similaire au fragment suivant :

```
myClip.nestedClip.gotoAndPlay(15);
myClip.someOtherClip.gotoAndStop("tweenIn");
```

Remarquez que le clip que vous manipulez (tel que `nestedClip`) apparaît juste avant l'action. Vous remarquerez également cette tendance dans les sections suivantes.

Votre accès n'est pas limité aux méthodes et aux propriétés prédéfinies des occurrences de la scène présentées dans les exemples précédents. Vous pouvez également définir une variable à l'intérieur d'un clip, comme dans le code suivant qui définit une variable dans le clip starClip :

```
starClip.speed = 1.1;
starClip.gravity = 0.8;
```

Si des variables de vitesse ou de gravité existaient déjà dans l'occurrence du clip starClip, les valeurs précédentes ont été remplacées dès la définition des nouvelles valeurs. Vous pouvez ajouter de nouvelles propriétés au clip starClip puisque la classe MovieClip a été définie avec le mot-clé `dynamic`. Le mot-clé `dynamic` indique que les objets basés sur la classe spécifiée (dans le cas présent, MovieClip) peuvent ajouter des propriétés dynamiques et y accéder pendant la période d'exécution. Pour plus d'informations sur l'instruction dynamic, reportez-vous à l'instruction dynamic dans le *Guide de référence du langage ActionScript 2.0*.

Ciblage dynamique d'occurrences et de contenu chargé

Vous pouvez également créer un objet à l'aide d'ActionScript, puis utiliser un chemin de cible. Par exemple, le code ActionScript suivant permet de créer un clip. Vous pouvez ensuite modifier la rotation de ce clip à l'aide d'ActionScript, comme dans l'exemple suivant :

Pour cibler une occurrence de clip créée de façon dynamique :

1. Créez un nouveau document FLA, puis enregistrez-le sous le nom **targetClip.fla**.
2. Insérez un nouveau calque et renommez-le **actions**.
3. Ajoutez le code ActionScript suivant dans l'image 1 du calque actions :

```
this.createEmptyMovieClip("rotateClip", this.getNextHighestDepth());
trace(rotateClip);
rotateClip._rotation = 50;
```

4. Choisissez Contrôe > Tester l'animation pour tester votre document.

   Vous savez que vous avez créé un clip par la présence de l'instruction `trace`, mais rien n'est visible sur la scène. Bien que vous ayez ajouté le code qui crée l'occurrence de clip, vous ne verrez rien sur la scène avant d'avoir ajouté un élément au clip. Vous pouvez par exemple charger une image dans le clip.
5. Revenez dans l'environnement de programmation, puis ouvrez le panneau Actions.
6. Tapez le code ActionScript suivant à la suite du code ajouté à l'étape 3 :

```
rotateClip.loadMovie("http://www.helpexamples.com/flash/images/
  image1.jpg");
```

   Ce code charge une image dans le clip rotateClip que vous avez créé. Vous ciblez l'occurrence rotateClip avec ActionScript.
7. Choisissez Contrôle > Tester l'animation pour tester votre document.

   Une image doit à présent s'afficher sur la scène avec une rotation de 50º dans le sens des aiguilles d'une montre.

Vous pouvez également cibler ou identifier des parties de fichiers SWF chargés dans un fichier SWF de base.

Pour identifier un fichier SWF chargé :

- Utilisez `_levelX`, où *X* est le numéro du niveau spécifié dans la fonction `loadMovie()` qui a chargé le fichier SWF.

  Par exemple, un fichier SWF chargé au niveau 99 a pour chemin cible `_level99`.
  Dans l'exemple suivant, un fichier SWF est chargé au niveau 99 et sa visibilité est définie sur `false` :

```
//Charge le SWF sur le niveau 99.
loadMovieNum("contents.swf", 99);
// Définit la visibilité du niveau 99 sur false.
loaderClip.onEnterFrame = function(){
  _level99._visible = false;
};
```

CONSEIL	Il est généralement recommandé d'éviter l'utilisation de niveaux si vous pouvez charger le contenu dans des clips à différents niveaux. La méthode `MovieClip.getNextHighestDepth()` vous permet de créer des occurrences de clip sur la scène de façon dynamique, sans qu'il soit nécessaire de vérifier s'il existe déjà une occurrence à une profondeur spécifique.

Définition de variables à l'aide d'un chemin

Vous pouvez définir des variables pour des occurrences imbriquées dans d'autres occurrences. Par exemple, pour définir une variable pour un formulaire placé dans un autre formulaire, utilisez le code suivant. L'occurrence submitBtn est à l'intérieur de `formClip` sur le scénario principal :

```
this.formClip.submitBtn.mouseOver = true;
```

Vous pouvez déclarer une méthode ou une propriété d'un objet particulier (tel qu'un clip ou un champ de texte) à l'aide de ce modèle. Par exemple, la propriété d'un objet serait

```
myClip._alpha = 50;
```

Domaine et ciblage

Lorsque vous imbriquez des occurrences, le clip qui contient un second clip est appelé *parent* de l'occurrence imbriquée. L'occurrence imbriquée est elle appelée occurrence enfant. La scène principale et le scénario principal sont eux-mêmes avant tout un clip et peuvent donc être ciblés en tant que tels. Pour plus d'informations sur le domaine, consultez la section « Variables et domaine », à la page 62.

Le code ActionScript vous permet de cibler des occurrences et des scénarios parents. Pour cibler le scénario actif, utilisez le mot-clé `this`. Par exemple, lorsque vous ciblez un clip appelé myClip situé sur le scénario principal, utilisez

```
this.myClip.
```

Vous pouvez même éventuellement abandonner le mot-clé `this` et utiliser simplement

```
myClip
```

Le mot-clé `this` facilite la compréhension et la cohérence du code. Pour plus d'informations sur les pratiques de programmation conseillées, consultez le Chapitre 17, « Recommandations et conventions de programmation pour ActionScript 2.0 », à la page 715.

Si vous suivez le clip pour l'un des fragments de code ci-dessus, `_level0.myClip` apparaît dans le panneau de sortie. Toutefois, si du code ActionScript est placé à l'intérieur du clip myClip alors que vous voulez cibler le scénario principal, vous devez cibler le parent du clip (c'est-à-dire la scène principale). Double-cliquez sur un clip et placez le code ActionScript suivant sur le scénario du clip :

```
trace("me: " + this);
trace("my parent: " + this._parent);
```

Testez le fichier SWF. Le message suivant s'affiche dans le panneau de sortie :

```
me: _level0.myClip
my parent: _level0
```

Il indique que vous avez ciblé le scénario principal. Vous pouvez utiliser un `parent` pour créer un chemin relatif vers un objet. Par exemple, si le clip `dogClip` est imbriqué dans le clip d'animation animalClip, l'instruction suivante de l'occurrence dogClip indique à animalClip d'arrêter l'animation :

```
this._parent.stop();
```

Si vous connaissez bien Flash et ActionScript, vous avez certainement remarqué que certains utilisent le domaine `_root`. Le domaine `_root` fait généralement référence au scénario principal du document Flash actif. Dans la mesure du possible, évitez d'utiliser le domaine `_root` et utilisez les chemins cibles relatifs au lieu de `_root`.

Si vous utilisez `_root` dans votre code, des erreurs risquent de survenir si vous chargez le fichier SWF dans un autre document Flash. Lors du chargement du fichier SWF dans un autre fichier SWF, le domaine `_root` du fichier chargé peut pointer vers le domaine racine du fichier SWF dans lequel il est chargé au lieu de faire référence à son propre domaine racine. L'opération peut donc entraîner des réactions inattendues, voire une défaillance totale de la fonctionnalité.

Utilisation du bouton Chemin cible

Il n'est pas toujours simple de savoir à quoi correspond un chemin cible donné ou quel chemin cible est nécessaire pour quel fragment de code. Si vous ciblez une occurrence située sur la scène, le bouton Chemin cible permet d'identifier le chemin qui y conduit.

Pour utiliser le bouton Chemin cible :

1. Dans le panneau Actions (Fenêtre > Actions), cliquez sur le bouton Insérer un chemin cible. Les clips de votre document actif s'affichent dans une boîte de dialogue.
2. Sélectionnez l'une des occurrences dans la liste.
3. Cliquez sur OK.
4. Le chemin cible de l'occurrence sélectionnée s'affiche dans la fenêtre de script.

Syntaxe à barre oblique

La syntaxe à barre oblique était utilisée dans Flash 3 et 4 pour indiquer le chemin cible d'un clip ou d'une variable. Cette syntaxe est prise en charge par ActionScript 1.0 dans Flash Player 7 et les versions antérieures, mais pas dans ActionScript 2.0 et Flash Player 7 ou 8.

L'utilisation de cette syntaxe est donc déconseillée, sauf lorsque vous n'avez pas le choix, par exemple lors de la création d'un contenu destiné spécifiquement à Flash Player 4 ou Flash Lite 1.1 (et versions antérieures). Pour plus d'informations sur Flash Lite, consultez la page du produit Flash Lite.

Eléments de ponctuation

Le langage de Flash comprend plusieurs *éléments de ponctuation*. Les plus courants sont les points-virgules (`;`), les deux-points (`:`), les parenthèses (`()`) et les accolades (`{}`). Chacun de ces éléments a une signification précise dans le langage de Flash et permet de définir des types de données, de clore des instructions ou de structurer le code ActionScript. Les sections suivantes abordent l'emploi de ces éléments de ponctuation dans votre code.

Pour plus d'informations sur les éléments de ponctuation, consultez les sections suivantes :

- « Points-virgules et deux points », à la page 94
- « Accolades », à la page 95
- « Parenthèses », à la page 99
- « Présentation des littéraux », à la page 99
- « Présentation des commentaires », à la page 101

Pour plus d'informations sur l'opérateur point (`.`) et les opérateurs d'accès tableau (`[]`), consultez la section « Utilisation des opérateurs point et d'accès au tableau », à la page 153. Pour plus d'informations sur les espaces blancs et le formatage du code, consultez la section « Mise en forme de la syntaxe ActionScript », à la page 750.

Points-virgules et deux points

Comme le montre les deux lignes de code suivantes, les instructions ActionScript se terminent par un point-virgule (;) :

```
var myNum:Number = 50;
myClip._alpha = myNum;
```

Vous pouvez cependant omettre ce caractère car le compilateur ActionScript suppose alors que chaque ligne de code représente une instruction distincte. Néanmoins, la meilleure pratique consiste à utiliser des points-virgules, car cela rend le code plus lisible. Lorsque vous cliquez sur le bouton de formatage automatique du panneau Actions ou de la fenêtre de script, des points-virgules sont ajoutés par défaut à la fin de vos instructions.

REMARQUE	L'ajout d'un point-virgule à la fin de chaque instruction permet de placer plusieurs instructions sur une même ligne, mais dans ce cas, la lecture de votre code est rendue plus difficile.

Les points-virgules sont également employés dans les boucles `for`. Comme l'illustre l'exemple suivant, le point-virgule permet de séparer les paramètres. L'exemple fait une boucle de 0 à 9, puis affiche chaque nombre dans le panneau de sortie :

```
var i:Number;
for (i = 0; i < 10; i++) {
  trace(i); // 0,1,...,9
}
```

Le caractère deux-points (:) est utilisé dans le code pour affecter des types de données aux variables. Pour affecter un type de données spécifique à un élément, spécifiez son type à l'aide d'une syntaxe utilisant le mot-clé `var` ainsi que deux-points, comme illustré ci-dessous :

```
// typage strict de variable ou objet
var myNum:Number = 7;
var myDate:Date = new Date();
// typage strict de paramètres
function welcome(firstName:String, myAge:Number) {
}
// typage strict de paramètre et de valeur renvoyée
function square(num:Number):Number {
  var squared:Number = num * num;
  return squared;
}
```

Vous pouvez déclarer le type des données des objets en fonction des classes intégrées (Button, Date, MovieClip, etc.) et des classes et interfaces que vous créez. Le code suivant crée un nouvel objet de type personnalisé Student :

```
var firstStudent:Student = new Student();
```

Vous pouvez également spécifier que les objets sont de type Function ou Void. Pour plus d'informations sur l'affectation du type de données, consultez le Chapitre 3, « Données et types de données », à la page 35.

Accolades

Les accolades ({}) permettent de regrouper des événements, des définitions de classe et des fonctions ActionScript dans des blocs. L'accolade d'ouverture doit être placée sur la même ligne que la déclaration.

REMARQUE

Vous pouvez également placer l'accolade d'ouverture sur la ligne qui suit la déclaration, mais il est conseillé de la placer sur la même ligne, pour plus de cohérence. Pour plus d'informations sur les accolades et les conventions de programmation, consultez le Chapitre 17, « Recommandations et conventions de programmation pour ActionScript 2.0 », à la page 715.

Placez les instructions entre accolades lorsqu'elles appartiennent à une structure de contrôle (par exemple `if..else` ou `for`), même lorsque cette dernière ne comporte qu'une seule instruction. Cette pratique conseillée permet d'éviter les erreurs dans le code ActionScript, telles que les oublis d'accolades. L'exemple suivant montre un code dont la forme est médiocre :

```
var numUsers:Number;
if (numUsers == 0)
  trace("no users found.");
```

Bien que ce code puisse être validé, sa présentation est médiocre dans la mesure où les accolades ne sont pas placées autour des instructions.

CONSEIL

Si vous cliquez sur le bouton Format automatique, des accolades seront ajoutées à cette instruction.

Dans ce cas, le fait d'ajouter une seconde instruction après l'instruction trace permet de l'exécuter, que la variable `numUsers` soit égale à 0 ou non, ce qui risque de déboucher sur des résultats imprévus. Pour cette raison, ajoutez des accolades de façon à obtenir le code suivant :

```
var numUsers:Number;
if (numUsers == 0) {
  trace("no users found");
}
```

L'exemple suivant crée un objet écouteur et une occurrence MovieClipLoader.

```
var imgUrl:String = "http://www.helpexamples.com/flash/images/image1.jpg";
this.createEmptyMovieClip("img_mc", 100);
var mclListener:Object = new Object();
mclListener.onLoadStart = function() {
  trace("starting");
};
mclListener.onLoadInit = function(target_mc:MovieClip):Void {
  trace("success");
};
mclListener.onLoadError = function(target_mc:MovieClip):Void {
  trace("failure");
};
var myClipl:MovieClipLoader = new MovieClipLoader();
myClipl.addListener(mclListener);
myClipl.loadClip(imgUrl, img_mc);
```

Le prochain exemple affiche un simple fichier de classe qui permet de créer un objet Student. Vous étudierez les fichiers de classe de manière approfondie dans le Chapitre 6, « Classes », à la page 197.

Pour utiliser des accolades dans un fichier ActionScript :

1. Choisissez Fichier > Nouveau, puis sélectionnez Fichier ActionScript.
2. Choisissez Fichier > Enregistrer sous et nommez le nouveau document **Student.as**.
3. Ajoutez le code ActionScript suivant dans le fichier AS.

```
// Student.as
class Student {
  private var _id:String;
  private var _firstName:String;
  private var _middleName:String;
  private var _lastName:String;

  public function Student(id:String, firstName:String,
  middleName:String, lastName:String) {
    this._id = id;
    this._firstName = firstName;
    this._middleName = middleName;
    this._lastName = lastName;
  }
  public function get firstName():String {
    return this._firstName;
  }
  public function set firstName(value:String):Void {
    this._firstName = value;
  }
  // ...
}
```

4. Enregistrez le fichier de classe.
5. Choisissez Fichier > Nouveau, puis cliquez sur Document Flash pour créer un nouveau fichier FLA.
6. Enregistrer le fichier FLA sous le nom **student_test.fla**.
7. Saisissez le code ActionScript suivant sur l'image 1 du scénario principal :

```
// student_test.fla
import Student;
var firstStudent:Student = new Student("cst94121", "John", "H.", "Doe");
trace(firstStudent.firstName); // John
firstStudent.firstName = "Craig";
trace(firstStudent.firstName); // Craig
```

8. Choisissez Fichier > Enregistrer pour enregistrer les modifications dans le fichier student_test.fla.
9. Choisissez Contrôle > Tester l'animation pour tester les fichiers FLA et AS.

Le prochain exemple montre l'emploi des accolades avec les fonctions.

Pour utiliser des accolades avec des fonctions :

1. Choisissez Fichier > Nouveau, puis Document Flash pour créer un nouveau fichier FLA.
2. Choisissez Fichier > Enregistrer sous et nommez le fichier **checkform.fla**.
3. Faites glisser une occurrence de composant Label du panneau Composants jusqu'à la scène.
4. Ouvrez l'inspecteur des propriétés (Fenêtre > Propriétés > Propriétés) et, l'occurrence du composant Label étant sélectionnée, saisissez le nom **status_lbl** dans le champ Nom de l'occurrence.
5. Saisissez **200** dans le champ W (largeur) pour définir la largeur du composant sur 200 pixels.
6. Faites glisser une occurrence du composant TextInput sur la scène et nommez-la **firstName_ti**.
7. Faites glisser une occurrence du composant Button sur la scène et nommez-la **submit_button**.

8. Sélectionnez l'image 1 du scénario et, dans le panneau Actions, ajoutez le code ActionScript suivant :

```
function checkForm():Boolean {
  status_lbl.text = "";
  if (firstName_ti.text.length == 0) {
    status_lbl.text = "Please enter a first name.";
    return false;
  }
  return true;
}
function clickListener(evt_obj:Object):Void {
  var success:Boolean = checkForm();
};
submit_button.addEventListener("click", clickListener);
```

9. Choisissez Fichier > Enregistrer pour enregistrer le document Flash.
10. Choisissez Contrôle > Tester l'animation pour tester le code dans l'environnement de programmation.

    Dans le fichier SWF, un message d'erreur apparaît si vous cliquez sur l'occurrence Button de la scène alors que le composant `firstName_ti` TextInput ne contient pas de texte. Cette erreur s'affiche dans le composant Label et signale à l'utilisateur qu'il doit saisir un prénom.

L'exemple suivant d'utilisation des accolades présente la création et la définition de propriétés à l'intérieur d'un objet. Dans cet exemple, les propriétés sont définies dans l'objet en plaçant les noms des variables entre des accolades (`{}`) :

```
var myObject:Object = {id:"cst94121", firstName:"John", middleName:"H.",
  lastName:"Doe"};
var i:String;
for (i in myObject) {
  trace(i + ": " + myObject[i]);
}
/*
  id: cst94121
  firstName: John
  middleName: H.
  lastName: Doe
*/
```

Vous pouvez également utiliser des accolades vides comme syntaxe raccourcie pour la fonction `new Object()`. Le code suivant crée par exemple une occurrence d'objet vide :

```
var myObject:Object = {};
```

CONSEIL	N'oubliez pas de vérifier que chaque accolade d'ouverture est bien associée à une accolade de fermeture.

Parenthèses

Comme l'illustrent les lignes de code suivantes, la définition d'une fonction dans ActionScript se fait en plaçant les paramètres entre parenthèses [()] :

```
function myFunction(myName:String, myAge:Number, happy:Boolean):Void {
  // Insérez votre code ici.
}
```

Lorsque vous appelez une fonction, vous incluez également tous les paramètres transmis à la fonction entre parenthèses, comme dans l'exemple suivant :

```
myFunction("Carl", 78, true);
```

Vous pouvez utiliser les parenthèses pour supplanter l'ordre de priorité d'ActionScript ou pour faciliter la compréhension de vos instructions ActionScript. Cela signifie que vous pouvez modifier l'ordre dans lequel les valeurs sont calculées en en plaçant certaines entre parenthèses, comme dans l'exemple suivant :

```
var computedValue:Number = (circleClip._x + 20) * 0.8;
```

Du fait de l'ordre de priorité, si vous n'utilisez pas de parenthèses ou que vous utilisez deux instructions distinctes, la multiplication est d'abord calculée et la première opération est donc 20 * 0,8. Le résultat, 16, est ensuite ajouté à la valeur actuelle de circleClip._x, puis affecté à la variable `computedValue`.

Si vous n'utilisez pas de parenthèses, vous devez ajouter une instruction pour évaluer l'expression, comme indiqué dans l'exemple suivant :

```
var tempValue:Number = circleClip._x + 20;
var computedValue:Number = tempValue * 0.8;
```

Comme pour les crochets et les accolades, vous devez vous assurer que chaque parenthèse d'ouverture est bien associée à une parenthèse de fermeture.

Présentation des littéraux

Un *littéral* est une valeur qui apparaît directement dans le code. Ce sont des valeurs constantes (qui ne changent pas) dans les documents Flash. `true`, `false`, `0`, `1`, `52` ou la chaîne « `foo` » en sont des exemples.

Voici quelques exemples de littéraux :

```
17
"hello"
-3
9.4
null
undefined
true
false
```

Les littéraux peuvent également être regroupés pour former des littéraux composés. Les littéraux de tableau sont placés entre crochets ([]) et utilisent la virgule (,) pour séparer les éléments du tableau. Un littéral de tableau permet donc d'initialiser un tableau. Les exemples suivants présentent deux tableaux initialisés par des littéraux de tableau. Vous pouvez utiliser l'instruction `new` et transmettre le littéral composé sous forme de paramètre au constructeur de classe Array, ou affecter directement les valeurs littérales lors de l'instanciation des occurrences d'une classe ActionScript intégrée.

```
// Utilisation de l'instruction new.
var myStrings:Array = new Array("alpha", "beta", "gamma");
var myNums:Array = new Array(1, 2, 3, 5, 8);

// affectation directe d'un littéral
var myStrings:Array = ["alpha", "beta", "gamma"];
var myNums:Array = [1, 2, 3, 5, 8];
```

Les littéraux permettent également d'initialiser un objet générique. Un objet générique est une occurrence de la classe Object. Les littéraux d'objet sont placés entre accolades ({}) et utilisent la virgule (,) pour séparer les propriétés de l'objet. Chaque propriété est déclarée avec le caractère deux-points (:), séparant le nom et la valeur de la propriété.

Vous pouvez créer un objet générique via l'instruction `new` et transmettre le littéral d'objet sous forme de paramètre au constructeur de classe Object ou affecter directement le littéral d'objet à l'occurrence déclarée. L'exemple suivant crée un objet générique et initialise l'objet avec trois propriétés, `propA`, `propB` et `propC`, de valeurs respectives `1`, `2` et `3`.

```
// Utilisation de l'instruction new.
var myObject:Object = new Object({propA:1, propB:2, propC:3});

// affectation directe d'un littéral
var myObject:Object = {propA:1, propB:2, propC:3};
```

Ne confondez pas un littéral de chaîne avec un objet String. Dans l'exemple suivant, la première ligne de code crée le littéral de chaîne `firstStr`, et la deuxième ligne de code, l'objet String `secondStr` :

```
var firstStr:String = "foo"
var secondStr:String = new String("foo")
```

Utilisez des littéraux de chaîne sauf si, pour optimiser les performances, vous avez spécifiquement besoin d'un objet String. Pour plus d'informations sur les chaînes, consultez la section « Présentation des chaînes et de la classe String », à la page 443.

Présentation des commentaires

Les commentaires permettent d'annoter votre code à l'aide de descriptions claires qui sont ignorées par le compilateur. Vous pouvez insérer ces commentaires dans votre code pour décrire sa fonction ou les données renvoyées au document. Les commentaires permettent de se souvenir des choix de programmation importants et sont extrêmement utiles pour toute personne qui lit et doit comprendre votre code. Les commentaires doivent expliquer clairement l'objectif du code et ne pas se contenter de le paraphraser. Toute section qui ne s'explique pas d'elle-même dans le code doit faire l'objet de commentaires.

Il est vivement recommandé d'utiliser des commentaires pour ajouter des notes aux scripts. Les commentaires documentent les décisions prises lors de la programmation, en expliquant le pourquoi et le comment. Ils facilitent la compréhension du code ActionScript. Par exemple, vous devez décrire toute solution de contournement dans les commentaires. Ainsi, vous (ou tout autre développeur) pourrez aisément localiser les sections de code concernées pour les mettre à jour ou les corriger. De même, lorsqu'un problème est corrigé par la suite dans la version suivante de Flash ou Flash Player, vous pouvez améliorer le code ActionScript en supprimant ce contournement devenu inutile.

Evitez de surcharger vos commentaires. Une ligne de signes égal (=) ou d'astérisques (*) créant un bloc ou une séparation autour du commentaire est un bon exemple de commentaire surchargé. Dans ce cas, utilisez des espaces blancs pour séparer les commentaires du code ActionScript. Si vous formatez le code ActionScript à l'aide du bouton Format automatique du panneau Actions ou de la fenêtre de script, l'espace blanc sera supprimé. N'oubliez pas de rajouter un espace blanc dans votre code, ou d'utiliser des lignes de commentaires uniques (//) pour conserver l'espacement. Il est ensuite plus facile de supprimer ces lignes après la mise en forme du code que d'essayer de déterminer où se trouvaient les espaces.

Avant le déploiement de votre projet, retirez du code tous les commentaires inutiles, tels que « Définition des variables x et y » ou autres commentaires évidents pour les autres développeurs. Si votre code ActionScript comporte trop de commentaires, envisagez la réécriture de certaines sections. La nécessité d'inclure de nombreux commentaires sur le fonctionnement du code ActionScript dénote généralement d'une programmation maladroite et peu intuitive.

Lorsque la coloration de la syntaxe est activée, les commentaires apparaissent en gris par défaut. Quelle que soit leur longueur, la taille du fichier exporté n'est pas affectée.
Les commentaires ne sont pas soumis aux règles de syntaxe ou de mots-clés d'ActionScript.

REMARQUE

L'emploi de commentaires est particulièrement important lorsque le code ActionScript est destiné à la formation. Ajoutez des commentaires à votre code si vous créez des exemples d'application dans le but de former les utilisateurs à la programmation Flash ou si vous rédigez des articles et des didacticiels sur ActionScript.

Commentaires sur une ligne

Ces commentaires sont limités à une ligne dans votre code. Vous pouvez alors commenter une seule ligne de code ou décrire brièvement l'objectif d'un fragment de code. Pour indiquer qu'une ligne ou portion de ligne est un commentaire, faites-la précéder de deux barres obliques (//), comme dans le code suivant :

```
// Le code suivant définit une variable locale pour l'âge.
var myAge:Number = 26;
```

Les commentaires sur une ligne portent généralement sur un petit fragment de code. Utilisez ce type de commentaires lorsque le contexte le permet. L'exemple suivant inclut un commentaire sur une ligne :

```
while (condition) {
  // Traiter la condition avec des instructions
}
```

Commentaires sur plusieurs lignes

Ces commentaires, également appelés blocs de commentaires, conviennent lorsque les commentaires comprennent plusieurs lignes. Les développeurs les utilisent pour décrire des fichiers, des structures de données, des méthodes et des descriptions de fichiers. Ces blocs sont généralement placés au début d'un fichier et avant ou à l'intérieur d'une méthode.

Pour créer un bloc de commentaires, entrez `/*` au début de la première ligne de commentaire, puis `*/` à la fin du bloc. Cette technique permet de créer des commentaires longs sans avoir à ajouter des // au début de chaque ligne. L'utilisation des // sur plusieurs lignes consécutives peut entraîner des problèmes en cas de modification des commentaires.

Le format d'un commentaire sur plusieurs lignes est le suivant.

```
/*
  The following ActionScript initializes variables used in the main and
  sub-menu systems. Variables are used to track what options are clicked.
*/
```

CONSEIL

Si vous placez les caractères de commentaire (/* et */) sur des lignes distinctes au début et à la fin du commentaire, vous pouvez les supprimer facilement en les faisant précéder de deux barres obliques (//) (par exemple, ///* et //*/). Cette fonction permet d'ajouter et de retirer facilement les commentaires de votre code.

En plaçant de gros pavés de code dans un bloc de commentaires, vous pouvez tester des sections spécifiques du script. Par exemple, lors de l'exécution du script suivant, le code intégré dans le bloc de commentaires n'est pas exécuté :

```
// Le code suivant s'exécute.
var x:Number = 15;
var y:Number = 20;

// Le code suivant est commenté et ne sera pas exécuté.
/*
// Crée un nouvel objet Date
var myDate:Date = new Date();
var currentMonth:Number = myDate.getMonth();
// Convertit le chiffre du mois en nom
var monthName:String = calcMonth(currentMonth);
var year:Number = myDate.getFullYear();
var currentDate:Number = myDate.getDate();
*/

// Le code ci-dessous s'exécute.
var namePrefix:String = "My name is";
var age:Number = 20;
```

CONSEIL

Il est recommandé de faire précéder chaque bloc de commentaires d'une ligne vierge.

Commentaires en fin de ligne

Ces commentaires permettent d'ajouter un commentaire dans une ligne de votre code. Ils apparaissent sur la même ligne que votre code ActionScript. Les développeurs les utilisent généralement pour décrire le contenu d'une variable ou une valeur renvoyée par une ligne du code ActionScript. Le format des commentaires en fin de ligne est le suivant :

```
var myAge:Number = 26; // Variable pour mon âge
trace(myAge); // 26
```

Déplacez les commentaires vers la droite, de sorte que les lecteurs les différencient bien du code. Dans la mesure du possible, alignez-les, comme illustré ci-dessous.

```
var myAge:Number = 28;            // mon âge
var myCountry:String = "Canada";  // mon pays
var myCoffee:String = "Hortons";  //mon café préféré
```

Si vous utilisez la fonction de formatage automatique (bouton Format automatique du panneau Actions), les commentaires en fin de ligne sont déplacés à la ligne suivante. Ajoutez ces commentaires après avoir formaté votre code ou changez-les de place après avoir utilisé la fonction de formatage automatique.

Commentaires dans les classes

Ces commentaires permettent de documenter le contenu de vos classes et de vos interfaces et facilitent la compréhension des développeurs. Faites débuter tous vos fichiers de classe par un commentaire fournissant le nom de classe, son numéro de version, sa date et votre droit de propriété. Par exemple, vous pouvez documenter votre classe de la façon suivante :

```
/**
  Pelican class
  version 1.2
  10/10/2005
  copyright Adobe Systems Incorporated
*/
```

Servez-vous de blocs de commentaires pour décrire les fichiers, les structures de données et les méthodes. Ces blocs sont généralement placés au début d'un fichier et avant ou à l'intérieur d'une méthode.

Un fichier d'interface ou de classe comporte généralement deux types de commentaire : des commentaires de documentation et des commentaires d'implémentation. Les commentaires de documentation décrivent le cahier des charges du code, mais pas son implémentation. Ils documentent les interfaces, les classes, les méthodes et les constructeurs. Les commentaires d'implémentation décrivent le code ou l'implémentation de sections spécifiques du code.

Incluez un commentaire de documentation par classe, interface ou membre, et placez-le directement avant la déclaration. Si vous devez faire des ajouts à vos commentaires de documentation, utilisez des commentaires d'implémentation (sous forme de blocs de commentaires ou de commentaires sur une ligne). Les commentaires d'implémentation suivent directement la déclaration.

Ces deux types de commentaires utilisent des séparateurs légèrement différents. Les commentaires de documentation sont séparés par `/**` et `*/`, tandis que les commentaires d'implémentation sont séparés par `/*` et `*/`.

CONSEIL	N'incluez pas les commentaires non directement relatifs à la classe en cours de lecture. Par exemple, n'incluez pas de commentaires décrivant le package correspondant.

Vous pouvez également utiliser des commentaires sur une ligne, des blocs de commentaires et des commentaires de fin de ligne dans les fichiers de classe. Pour plus d'informations sur ces types de commentaires, consultez les sections suivantes :

- « Commentaires sur une ligne », à la page 102
- « Commentaires sur plusieurs lignes », à la page 102
- « Commentaires en fin de ligne », à la page 103

Présentation des constantes et des mots-clés

Les constantes et les mots-clés sont la base de la syntaxe ActionScript. Les constantes sont des propriétés dont la valeur est fixe et non modifiable. Elles ne changent donc jamais, même pendant l'exécution de l' application.

Flash comprend plusieurs constantes prédéfinies qui peuvent simplifier le développement d'application. Vous pouvez observer un exemple de constantes dans la classe Key, qui comprend de nombreuses propriétés, comme `Key.ENTER` ou `Key.PGDN`. Lorsque vous utilisez des constantes, vous n'avez pas besoin de vous souvenir que les valeurs de code des touches Entrée et Page suivante sont 13 et 34. Ces constantes ne simplifient pas seulement le développement et le débogage, mais facilitent la compréhension de votre code par vos collègues développeurs.

Les mots-clés utilisés par ActionScript permettent d'exécuter des types d'actions spécifiques. Il s'agit également de mots réservés et vous ne pouvez donc pas les utiliser comme identifiants (noms de variable, de fonction, d'étiquette, etc.). Voici quelques exemples de mots-clés réservés : `if`, `else`, `this`, `function` et `return`.

Pour plus d'informations sur les constantes et les mots-clés, consultez les sections suivantes :

- « Utilisation des constantes », à la page 105
- « Présentation des mots-clés », à la page 108
- « Présentation des mots réservés », à la page 109

Pour plus d'informations sur les objets et les propriétés, consultez la section « Type de données Object », à la page 43. Vous trouverez la liste des constantes du langage ActionScript (telles que `false` et `NaN`) dans la catégorie Eléments du langage ActionScript > Constantes du *Guide de référence du langage ActionScript 2.0*.

Utilisation des constantes

Les constantes sont des propriétés de valeurs fixes non modifiables. En d'autres termes, elles ne changent jamais, même pendant l'exécution de l'application. Le langage ActionScript contient un grand nombre de constantes prédéfinies. Par exemple, les constantes `BACKSPACE`, `ENTER`, `SPACE` et `TAB` sont des propriétés de la classe Key qui font référence aux touches du clavier. La constante `Key.TAB` a toujours la même signification : elle indique la touche Tab du clavier. Les constantes sont très utiles pour comparer des valeurs et utiliser des valeurs qui ne changent pas dans votre application.

Pour savoir si un utilisateur appuie sur la touche Entrée, vous pouvez utiliser l'instruction suivante :

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.getCode() == Key.ENTER) {
    trace("Are you ready to play?");
  }
};
Key.addListener(keyListener);
```

Pour que le code ActionScript précédent fonctionne, il peut être nécessaire de désactiver les raccourcis clavier dans l'environnement de programmation. Dans le menu principal, choisissez Contrôle > Tester l'animation puis, avec un aperçu du fichier SWF dans le lecteur, choisissez Contrôle > Désactiver les raccourcis clavier dans la fenêtre d'aperçu du fichier SWF.

Flash ne vous permet pas de créer vos propres valeurs de constantes, sauf lorsque vous créez vos propres classes personnalisées avec des variables de membre privé. Vous ne pouvez pas créer de variable en « lecture seule » dans Flash.

Les variables doivent être en minuscules ou comprendre un mélange de minuscules et majuscules. Toutefois, les constantes (variables qui ne changent pas) doivent être en majuscules. Séparez les mots par un caractère de soulignement, comme dans l'exemple suivant de code ActionScript :

```
var BASE_URL:String = "http://www.adobe.com"; // constante
var MAX_WIDTH:Number = 10;                    // constante
```

Tapez les constantes statiques en majuscules et séparez les mots par des caractères de soulignement. Ne codez pas directement les constantes numériques, sauf si leur valeur est 1, 0 ou -1 que vous pouvez utiliser dans le cadre d'une boucle `for` en tant que valeur de compteur.

Vous pouvez utiliser des constantes pour les situations où vous devez faire référence à une propriété dont la valeur ne change jamais. Ceci permet d'identifier les erreurs typographiques qui risquent de passer inaperçues avec des littéraux. Ceci vous permet également de centraliser la modification des valeurs. Pour plus d'informations sur les littéraux, consultez la section « Présentation des littéraux », à la page 99.

Par exemple, la définition de classe de l'exemple suivant crée trois constantes qui respectent la convention d'appellation utilisée par ActionScript 2.0.

Pour utiliser des constantes dans une application :

1. Choisissez Fichier > Nouveau, puis Fichier ActionScript pour créer un fichier AS.
2. Nommez ce fichier **ConstExample.as**.
3. Saisissez le code suivant dans la fenêtre de script :

```
class ConstExample {
  public static var EXAMPLE_STATIC:String = "Global access";
  public var EXAMPLE_PUBLIC:String = "Public access";
  private var EXAMPLE_PRIVATE:String = "Class access";
}
```

   La propriété `EXAMPLE_STATIC` est statique, c'est-à-dire qu'elle s'applique à la classe entière et non à une occurrence particulière de la classe. Pour accéder à une propriété statique d'une classe, vous devez utiliser le nom de la classe à la place du nom de l'occurrence. Vous ne pouvez pas accéder à une propriété statique par l'intermédiaire d'une occurrence de classe.
4. Créez un nouveau document Flash, puis enregistrez-le sous le nom **const.fla**.
5. Dans le panneau Actions, entrez le code suivant dans l'image 1 du scénario :

```
trace(ConstExample.EXAMPLE_STATIC); // résultat : Global access
```

   Lorsque la propriété `EXAMPLE_STATIC` est déclarée statique, utilisez ce code pour accéder à sa valeur.
6. Choisissez Contrôle > Tester l'animation pour tester votre document.

   La mention `Global access` s'affiche dans le panneau de sortie.
7. Dans le panneau Actions, saisissez le code suivant après celui que vous avez ajouté à l'étape 5.

```
trace(ConstExample.EXAMPLE_PUBLIC); // Erreur
trace(ConstExample.EXAMPLE_PRIVATE); // erreur
```

8. Choisissez Contrôle > Tester l'animation pour tester votre document.

   Les propriétés `EXAMPLE_PUBLIC` et `EXAMPLE_PRIVATE` ne sont pas statiques. Lorsque vous tentez d'accéder à leurs valeurs à travers la classe, le message d'erreur suivant apparaît :

```
The property being referenced does not have the static attribute.
```

   Pour accéder à une propriété non statique, vous devez passer par l'occurrence de la classe. La propriété `EXAMPLE_PUBLIC` étant publique, le code peut y accéder en dehors de la définition de la classe.
9. Dans le panneau Actions, supprimez l'instruction `trace` que vous avez ajoutée lors des étapes 5 et 7.

10. Tapez le code suivant dans le panneau Actions :

```
var myExample:ConstExample = new ConstExample();
trace(myExample.EXAMPLE_PUBLIC); // résultat : Public access
```

Ce code crée l'occurrence myExample et accède à la propriété `EXAMPLE_PUBLIC`.

11. Choisissez Contrôle > Tester l'animation pour tester votre document.

La mention `Public access` s'affiche dans le panneau de sortie.

12. Dans le panneau Actions, supprimez l'instruction trace que vous avez ajoutée à l'étape 10.

13. Tapez le code suivant dans le panneau Actions.

```
trace(myExample.EXAMPLE_PRIVATE); // erreur
```

La propriété `EXAMPLE_PRIVATE` étant privée, elle n'est accessible qu'à l'intérieur de la définition de la classe.

14. Choisissez Contrôle > Tester l'animation pour tester votre document.

La mention `The member is private and cannot be accessed` s'affiche dans le panneau de sortie.

Pour plus d'informations sur les classes intégrées et la création de classes personnalisées, consultez le Chapitre 6, « Classes », à la page 197.

Présentation des mots-clés

Les mots-clés sont des mots d'ActionScript qui ont une fonction spéciale. Par exemple, le mot-clé `var` permet de déclarer une variable. Le mot-clé `var` est présenté dans le code suivant :

```
var myAge:Number = 26;
```

Il s'agit de mots réservés dont la signification est particulière : par exemple, le mot-clé `class` permet de définir une nouvelle classe ActionScript et le mot-clé `var` de déclarer des variables locales. Voici d'autres exemples de mots-clés réservés : `if`, `else`, `this`, `function` et `return`.

Les mots-clés ne peuvent pas servir d'identifiants (noms de variable, fonction ou étiquette) et il est préférable de ne pas les utiliser ailleurs que dans vos fichiers FLA pour d'autres éléments (tels que des noms d'occurrences). Vous avez déjà largement utilisé le mot-clé `var`, en particulier si vous avez lu le Chapitre 3, « Données et types de données », à la page 35. ActionScript réserve les mots du langage à un usage spécifique. De ce fait, vous ne pouvez pas les utiliser comme identifiants (noms de variable, de fonction ou d'étiquette). Vous trouverez la liste de ces mots-clés dans la section « Présentation des mots réservés », à la page 109.

Présentation des mots réservés

Les mots réservés ne peuvent pas servir d'identifiants dans votre code car leur utilisation est réservée à ActionScript. Ces mots réservés incluent les *mots-clés*, correspondant à des instructions ActionScript, et les mots réservés pour une utilisation future. Cela signifie que vous ne pouvez pas les utiliser comme noms de variables, d'occurrences, de classes personnalisées, etc., sans provoquer des problèmes techniques.

D'autres termes, qui ne sont pas des mots réservés, ne doivent pas non plus servir d'identifiants (par exemple les noms de variables ou d'occurrences) dans votre code ActionScript. Ces mots sont utilisés par les classes intégrées qui composent le langage ActionScript, ils sont appelés éléments de langage. N'utilisez donc pas non plus les noms des propriétés, méthodes, classes, interfaces, classes de composants et interfaces comme identifiants de variables, classes ou occurrences dans votre code.

Pour plus d'informations sur les mots-clés réservés susceptibles de provoquer des erreurs dans vos scripts et les mots-clés protégés pour une utilisation future par ActionScript ou la spécification ECMAScript (ECMA-262) édition 4, consultez la section Utilisation restreinte des mots réservés et des éléments de langage à la page 719.

Pour connaître les noms des élements de langage, consultez le *Guide de référence du langage ActionScript 2.0*.

Présentation des instructions

Une *instruction* est une action spécifique que vous demandez au fichier FLA d'exécuter. Par exemple, vous pouvez utiliser une instruction conditionnelle pour déterminer si quelque chose est vrai ou existe. Votre code peut ensuite exécuter les actions spécifiées, telles que des fonctions ou des expressions, selon si la condition est vraie ou non.

Par exemple, l'instruction `if` est une instruction conditionnelle qui évalue une condition afin de déterminer l'action suivante du code.

```
// instruction if
if (condition) {
  // instructions ;
}
```

Dans un autre exemple, l'instruction `return` renvoie un résultat sous forme de valeur de la fonction dans laquelle elle s'exécute.

Il existe différents moyens pour formater ou rédiger du code ActionScript. Chacun choisit sa propre utilisation de la syntaxe et espace ou place différemment ses instructions ou accolades (`{}`) dans le code. Il existe malgré tout des consignes générales à respecter lors de l'écriture d'un code ActionScript bien présenté.

Ne placez qu'une seule instruction par ligne pour améliorer la lisibilité de votre code ActionScript. Vous trouverez ci-dessous des exemples d'utilisation conseillée ou déconseillée des instructions :

```
theNum++;       // recommandé
theOtherNum++;  // recommandé
aNum++; anOtherNum++;  // déconseillé
```

Affectez les variables en tant qu'instructions distinctes. Examinez l'exemple de code ActionScript suivant :

```
var myNum:Number = (a = b + c) + d;
```

Ce code imbrique une affectation dans le code, ce qui devient difficile à lire. Si l'affectation des variables s'effectue par des instructions distinctes, le code devient plus lisible, comme dans l'exemple suivant :

```
var a:Number = b + c;
var myNum:Number = a + d;
```

Les sections suivantes présentent la composition d'instructions spécifiques dans ActionScript. Pour plus d'informations sur la rédaction et la mise en forme des événements, consultez le Chapitre 9, « Gestion d'événements », à la page 311.

Pour plus d'informations sur chaque instruction, consultez les sections suivantes :

- « Présentation des instructions composées », à la page 110
- « Présentation des conditions », à la page 111
- « Répétition d'actions à l'aide de boucles », à la page 122

Présentation des instructions composées

Une instruction composée contient plusieurs instructions placées entre des accolades ({}). Ces instructions composées peuvent contenir n'importe quel type d'instruction ActionScript. Voici un exemple d'instruction composée typique :

Les instructions placées entre accolades sont en retrait par rapport à l'instruction composée, comme dans l'exemple de code ActionScript suivant :

```
var a:Number = 10;
var b:Number = 10;
if (a == b) {
  // Ce code est en retrait.
  trace("a == b");
  trace(a);
  trace(b);
}
```

Cette instruction composée contient plusieurs instructions mais se comporte comme une seule instruction dans votre code ActionScript. L'accolade d'ouverture est placée à la fin de l'instruction composée. L'accolade de fermeture débute une ligne et s'aligne sur le début de l'instruction composée.

Pour plus d'informations sur l'utilisation des accolades, consultez la section « Accolades », à la page 95.

Présentation des conditions

Les conditions permettent de déterminer si une chose est vraie ou existe. Vous pouvez ensuite éventuellement répéter une action (à l'aide de boucles) ou exécuter des actions définies, par exemple des fonctions ou des expressions, selon l'état d'une condition (vraie ou fausse). Par exemple, vous pouvez déterminer si une certaine variable est définie ou a une certaine valeur, puis exécuter un bloc de code en fonction du résultat. De même, vous pourriez modifier les graphiques de votre document Flash selon l'heure de l'horloge système de l'utilisateur ou les conditions météorologiques de sa région.

Pour exécuter une action en fonction d'une condition ou pour la répéter (boucle), vous pouvez utiliser les instructions `if`, `else`, `else if`, `for`, `while`, `do while`, `for..in` ou `switch`.

Pour plus d'informations sur les conditions disponibles, et leur rédaction, consultez les sections suivantes :

- « Rédaction des conditions », à la page 112
- « Utilisation de l'instruction if », à la page 112
- « Utilisation de l'instruction if..else », à la page 113
- « Utilisation de l'instruction if..else if », à la page 115
- « Utilisation de l'instruction switch », à la page 116
- « Utilisation des instructions try..catch et try..catch..finally », à la page 118
- « Opérateur conditionnel et syntaxe de remplacement », à la page 121

Rédaction des conditions

Les instructions qui vérifient l'état d'une condition commencent par le terme `if`. Si cette condition renvoie `true`, ActionScript exécute l'instruction suivante. Si la condition renvoie (`false`), ActionScript passe à l'instruction suivante à l'extérieur du bloc de code.

CONSEIL	Pour optimiser les performances de votre code, vérifiez d'abord les conditions les plus probables.

Les instructions suivantes testent trois conditions. Le terme `else if` spécifie d'autres tests à effectuer si les conditions précédentes sont false.

```
if ((passwordTxt.text.length == 0) || (emailTxt.text.length == 0)) {
  gotoAndStop("invalidLogin");
}  else if (passwordTxt.text == userID){
  gotoAndPlay("startProgram");
}
```

Dans ce fragment de code, si la longueur des champs passwordTxt ou emailTxt est 0 (lorsque l'utilisateur ne les a pas renseignés par exemple), le document Flash redirige vers l'étiquette d'image `invalidLogin`. Si les champs passwordTxt et emailTxt contiennent des valeurs et que leur contenu correspond à la variable `userID`, le fichier SWF est redirigé vers l'étiquette d'image `startProgram`.

Pour vérifier une condition parmi d'autres, utilisez l'instruction `switch`, plutôt que plusieurs instructions `else if`. Pour plus d'informations sur les instructions `switch`, consultez la section « Utilisation de l'instruction switch », à la page 116.

Pour apprendre à rédiger les différents types de conditions dans vos applications ActionScript, consultez les sections suivantes.

Utilisation de l'instruction if

L'instruction `if` permet d'exécuter une suite d'instructions si une certaine condition est vraie (true).

```
// instruction if
if (condition) {
  // instructions ;
}
```

Vous utiliserez fréquemment les instructions `if` lorsque vous travaillerez sur un projet Flash. Par exemple, si vous construisez un site Flash réclamant l'identification des utilisateurs pour l'accès à certaines pages, utilisez une instruction `if` pour vérifier si l'utilisateur a renseigné les champs Nom d'utilisateur et Mot de passe.

Si le nom d'utilisateur et le mot de passe doivent être validés via une base de données externe, vous souhaiterez probablement vous assurer que la combinaison nom d'utilisateur/mot de passe saisie par l'utilisateur correspond à un enregistrement de la base de données. Vous souhaiterez également vérifier que l'utilisateur est bien autorisé à accéder à la partie du site spécifiée.

Si vous créez des animations à l'aide de script dans Flash, vous pouvez utiliser l'instruction `if` pour vérifier qu'une occurrence de la scène reste dans les limites de celle-ci. Par exemple, si une balle descend sous l'axe des y, il peut être nécessaire de déceler le moment où elle heurte le bord inférieur de la scène pour modifier sa direction et la faire rebondir.

Pour utiliser une instruction if :

1. Choisissez Fichier > Nouveau, puis Document Flash.
2. Sélectionnez l'image 1 du scénario et saisissez le code ActionScript suivant dans le panneau Actions :

```
// Création d'une chaîne pour gérer les heures AM et PM
var amPm:String = "AM";
// aucun paramètre ne renvoie de date, la date/heure en cours est
// renvoyée
var current_date:Date = new Date();
// si l'heure en cours est supérieure ou égale à 12, définition de
// la chaîne amPm sur « PM ».
if (current_date.getHours() >= 12) {
  amPm = "PM";
}
trace(amPm);
```

3. Choisissez Contrôle > Tester l'animation pour tester le code ActionScript.

   Dans ce code, vous créez une chaîne qui gère la mention `AM` ou `PM` en fonction de l'heure de la journée. Si l'heure est supérieure ou égale à 12, la chaîne `amPM` est définie sur `PM`. Enfin, vous suivez la chaîne `amPm` et lorsque l'heure devient supérieure ou égale à 12, la mention `PM` s'affiche. Dans le cas contraire, la mention `AM` apparaît.

Utilisation de l'instruction if..else

L'instruction conditionnelle `if..else` permet de tester une condition, puis d'exécuter un bloc de code lorsque cette condition est positive et d'en exécuter un autre dans le cas contraire.

Par exemple, le code suivant vérifie si la valeur de `x` est supérieure à 20 et génère une instruction `trace()` dans l'affirmative ou une autre instruction `trace()` dans le cas contraire :

```
if (x > 20) {
  trace("x is > 20");
} else {
  trace("x is <= 20");
}
```

Si vous ne souhaitez pas exécuter un autre bloc de code, vous pouvez utiliser l'instruction `if` sans l'instruction `else`.

L'instruction `if..else` est similaire à l'instruction `if`. Par exemple, si vous utilisez l'instruction `if` pour vérifier que le nom et le mot de passe saisis par les utilisateurs correspondent bien à une valeur stockée dans la base de données, vous pouvez ensuite rediriger les utilisateurs selon si leurs identifiants sont corrects. En cas d'identification valide, l'utilisateur peut être dirigé vers une page d'accueil par le bloc `if`. Si l'identification échoue, vous pouvez rediriger l'utilisateur vers le formulaire d'identification et afficher un message d'erreur via le bloc `else`.

Pour utiliser une instruction if..else dans un document :

1. Sélectionnez Fichier > Nouveau, puis sélectionnez Document Flash pour créer un nouveau fichier FLA.
2. Sélectionnez l'image 1 du scénario et saisissez le code ActionScript suivant dans le panneau Actions :

```
// création d'une chaîne chargée de gérer la mention AM/PM en fonction
// de l'heure de la journée.
var amPm:String;
// aucun paramètre ne renvoie de date, la date/heure en cours est
// renvoyée.
var current_date:Date = new Date();
// si l'heure en cours est supérieure ou égale à 12, définition de
// la chaîne amPm sur « PM ».
if (current_date.getHours() >= 12) {
  amPm = "PM";
} else {
  amPm = "AM";
}
trace(amPm);
```

3. Choisissez Contrôle > Tester l'animation pour tester le code ActionScript.

   Dans ce code, vous créez une chaîne qui gère la mention `AM` ou `PM` en fonction de l'heure de la journée. Si l'heure est supérieure ou égale à 12, la chaîne `amPM` est définie sur `PM`. Enfin, vous suivez la chaîne `amPm` et lorsque l'heure devient supérieure ou égale à 12, la mention `PM` s'affiche. La mention `AM` s'affiche dans le panneau de sortie.

Utilisation de l'instruction if..else if

L'instruction conditionnelle `if..else if` permet de tester plusieurs conditions. L'instruction `if..else if` utilise la syntaxe suivante :

```
// instruction else-if
if (condition) {
  // instructions ;
} else if (condition) {
  // instructions ;
} else {
  // instructions ;
}
```

Vous pouvez utiliser un bloc `if..else if` dans vos projets Flash lorsque vous voulez vérifier une série de conditions. Par exemple, si vous souhaitez afficher à l'écran une image qui diffère selon l'heure à laquelle l'utilisateur se connecte, vous pouvez créer une suite d'instructions `if` qui détermine s'il s'agit du matin, de l'après-midi, du soir ou de la nuit. Vous pouvez ensuite afficher l'image appropriée.

Le code suivant regarde non seulement si la valeur de `x` est supérieure à 20, mais également si la valeur de `x` est négative :

```
if (x > 20) {
  trace("x is > 20");
} else if (x < 0) {
  trace("x is negative");
}
```

Pour utiliser une instruction if..else if dans un document :

1. Choisissez Fichier > Nouveau, puis Document Flash.
2. Sélectionnez l'image 1 du scénario et saisissez le code ActionScript suivant dans le panneau Actions :

```
var now_date:Date = new Date();
var currentHour:Number = now_date.getHours();
// si l'heure est inférieure à 11 H (AM)...
if (currentHour < 11) {
  trace("Good morning");
  // sinon.. si l'heure est inférieure à 15 H (3PM)...
} else if (currentHour < 15) {
  trace("Good afternoon");
  // sinon.. s'il n'est pas encore 20 H (8PM)...
} else if (currentHour < 20) {
  trace("Good evening");
  // sinon, l'heure est comprise entre 20 H (8PM) et 23H59 (11:59PM)
} else {
  trace("Good night");
}
```

3. Choisissez Contrôle > Tester l'animation pour tester le code ActionScript.

   Dans ce code, vous créez une chaîne appelée `currentHour` qui gère le nombre de l'heure en cours (par exemple, s'il est 6:19 pm, `currentHour` gère le nombre `18`). Pour obtenir l'heure en cours, utilisez la méthode `getHours()` de la classe Date. Vous utilisez ensuite l'instruction `if..else if` pour envoyer les informations au panneau de sortie, selon le nombre renvoyé. Pour plus d'informations, consultez les commentaires du fragment de code précédent.

Utilisation de l'instruction switch

L'instruction `switch` crée une structure arborescente pour des instructions ActionScript. Comme pour l'instruction `if`, l'instruction `switch` teste une condition et exécute des instructions si cette condition renvoie la valeur `true` (vrai).

Dans le cadre d'une instruction `switch`, l'instruction `break` force Flash à ignorer le reste des instructions dans le bloc de code de ce cas et passe à la première instruction qui suit l'instruction `switch` qui l'encadre. Si le bloc du cas ne contient pas d'instruction « break », une condition appelée « fall through » (passe au cas suivant) survient. Dans ctte situation, la première instruction du cas suivant s'exécute également jusqu'à ce que survienne une instruction « break » ou que l'instruction `switch` se termine. Ce comportement est illustré dans l'exemple suivant où la première instruction du cas ne contenant pas d'instruction break, les deux blocs de code des deux premiers cas (A et B) s'exécutent.

Toutes les instructions `switch` doivent inclure un cas `default`. Le cas `default` doit toujours être le dernier cas d'une instruction `switch` et comprendre également une instruction `break` afin d'éviter l'apparition d'une erreur « passer au cas suivant » (fall-through) en cas d'ajout d'un autre cas. Par exemple, si la condition de l'exemple suivant donne A, les deux instructions pour les cas A et B s'exécutent, dans la mesure où le cas A ne comporte pas d'instruction `break`. Lorsqu'un cas ne comporte pas d'instruction `break`, remplacez ce dernier par un commentaire, comme indiqué dans l'exemple suivant, après le cas A. Lorsque vous rédigez des instructions `switch`, utilisez le format suivant :

```
switch (condition) {
case A :
  // instructions
  // falls through
case B :
  // instructions
  break;
case Z :
  // instructions
  break;
default :
  // instructions
  break;
}
```

Pour utiliser une instruction switch dans un document :

1. Choisissez Fichier > Nouveau, puis Document Flash.
2. Sélectionnez l'image 1 du scénario et saisissez le code ActionScript suivant dans le panneau Actions :

```
var listenerObj:Object = new Object();
listenerObj.onKeyDown = function() {
  // Utilisation de la méthode String.fromCharCode() pour renvoyer une
  chaîne.
  switch (String.fromCharCode(Key.getAscii())) {
  case "A" :
    trace("you pressed A");
    break;
  case "a" :
    trace("you pressed a");
    break;
  case "E" :
  case "e" :
    /* E doesn't have a break statement, so this block executes if you
  press e or E. */
    trace("you pressed E or e");
    break;
  case "I" :
  case "i" :
    trace("you pressed I or i");
    break;
  default :
    /* If the key pressed isn't caught by any of the above cases,
  execute the default case here. */
    trace("you pressed some other key");
  }
};
Key.addListener(listenerObj);
```

3. Choisissez Contrôle > Tester l'animation pour tester le code ActionScript.

   Tapez des lettres au clavier, y compris les lettres a, e ou i. Lorsque vous appuyez sur ces trois touches, les instructions `trace` apparaissent dans le code ActionScript précédent. La ligne de code crée un nouvel objet que vous utilisez comme écouteur de la classe Key. Vous utilisez cet objet pour notifier l'événement `onKeyDown()` lorsque l'utilisateur appuie sur une touche. La méthode `Key.getAscii()` renvoyant le code ASCII de la dernière touche pressée ou relâchée par l'utilisateur, vous n'avez pas besoin d'utiliser la méthode `String.fromCharCode()` pour renvoyer une chaîne contenant les caractères représentés par les valeurs ASCII dans les paramètres. Le cas « E » ne présentant pas d'instruction `break`, le bloc s'exécute si l'utilisateur appuie sur la touche *e* ou *E*. Si la touche pressée ne correspond à aucun des trois premiers cas, le cas default s'exécute.

Utilisation des instructions try..catch et try..catch..finally

L'utilisation des blocs `try..catch..finally` vous permet d'ajouter un traitement des erreurs dans vos applications Flash. Les mots-clés `try..catch..finally` vous permettent d'entourer un bloc de code dans lequel une erreur peut survenir et de réagir à cette erreur. Lorsqu'un code figurant dans le bloc `try` lance une erreur (via l'instruction `throw`), le contrôle est transmis au bloc `catch` lorsque ce dernier existe. Le contrôle est ensuite transmis au bloc de code `finally` s'il en existe un. Le bloc `finally` facultatif s'exécute toujours, qu'une erreur ait été lancée ou non.

Si le code figurant dans le bloc `try` ne lance pas d'erreur (s'il se termine normalement), le code du bloc `finally` est tout de même exécuté.

REMARQUE	Le bloc finally s'exécute même si le bloc try se termine avec une instruction return.

Rédigez les instructions `try..catch` et `try..catch..finally` à l'aide du format suivant :

```
// try-catch
try {
  // instructions
} catch (myError) {
  // instructions
}

//try-catch-finally
try {
  // instructions
} catch (myError) {
  // instructions
} finally {
  // instructions
}
```

Chaque fois que votre code lance une erreur, vous pouvez rédiger des gestionnaires personnalisés pour réagir à cette défaillance en douceur et prendre les mesures appropriées. Vous pouvez tenter de charger des données externes depuis un service Web ou un fichier texte ou afficher un message d'erreur à l'utilisateur. Vous pouvez même utiliser le bloc `catch` pour tenter de vous connecter à un service Web qui signale l'erreur à l'administrateur de sorte qu'il puisse vérifier le bon fonctionnement de l'application.

Pour utiliser le bloc try..catch..finally pour valider les données avant de diviser certains nombres :

1. Choisissez Fichier > Nouveau, puis Document Flash.
2. Sélectionnez l'image 1 du scénario et saisissez le code ActionScript suivant dans le panneau Actions :

```
var n1:Number = 7;
var n2:Number = 0;
try {
  if (n2 == 0) {
    throw new Error("Unable to divide by zero");
  }
  trace(n1/n2);
} catch (err:Error) {
  trace("ERROR! " + err.toString());
} finally {
  delete n1;
  delete n2;
}
```

3. Sélectionnez Contrôle > Tester l'animation pour tester le document.
4. Le panneau de sortie affiche le message `Unable to divide by zero` (Division par zéro impossible).
5. Revenez dans l'environnement de programmation et modifiez la ligne de code suivante :

   ```
   var n2:Number = 0;
   ```

   en

   ```
   var n2:Number = 2;
   ```

6. Choisissez Contrôle > Tester l'animation pour tester de nouveau le document.

   Si la valeur de `n2` est égale à zéro, une erreur est lancée et interceptée par le bloc `catch` qui affiche un message dans le panneau de sortie. Si la valeur de `y` est différente de zéro, le panneau de sortie présente le résultat de `n1` divisé par `n2`. Le bloc `finally` s'exécute dans tous les cas et supprime les valeurs des variables `n1` et `n2` du document Flash.

Vous n'êtes pas limité au lancement de nouvelles occurrences de la classe Error lorsqu'une erreur survient. Vous pouvez également étendre cette classe pour créer vos propres erreurs personnalisées, comme l'illustre l'exemple suivant.

Pour créer une erreur personnalisée :

1. Choisissez Fichier > Nouveau et créez un fichier ActionScript.
2. Choisissez Fichier > Enregistrer sous et nommez le fichier **DivideByZeroException.as**.
3. Saisissez le code ActionScript suivant dans la fenêtre de script :

```
// Dans DivideByZeroException.as:
class DivideByZeroException extends Error {
   var message:String = "Divide By Zero error";
}
```

4. Enregistrez le fichier ActionScript.
5. Créez un document Flash nommé **exception_test.fla** dans le même répertoire que le fichier ActionScript et enregistrez-le.
6. Dans le panneau Actions, saisissez le code ActionScript suivant sur l'image 1 du scénario principal :

```
var n1:Number = 7;
var n2:Number = 0;
try {
  if (n2 == 0) {
    throw new DivideByZeroException();
  } else if (n2 < 0) {
    throw new Error("n2 cannot be less than zero");
  } else {
    trace(n1/n2);
  }
} catch (err:DivideByZeroException) {
  trace(err.toString());
} catch (err:Error) {
  trace("An unknown error occurred; " + err.toString());
}
```

7. Enregistrez le document Flash, puis choisissez Contrôle > Tester l'animation pour tester le fichier dans l'environnement de test.

   La valeur de `n2` étant égale à 0, Flash renvoie votre classe d'erreur personnalisée DivideByZeroException et affiche le message `Divide By Zero error` dans le panneau de sortie. Si, à la ligne deux, vous modifiez la valeur de `n2` de `0` en `-1`, et testez de nouveau le document Flash, le message `An unknown error occurred; n2 cannot be less than zero` (Erreur inconnue. n2 ne peut pas être inférieur à zéro) apparaît dans le panneau de sortie. La saisie de n'importe quelle valeur `n2` supérieure à `0` entraîne l'affichage du résultat de la division dans le panneau de sortie. Pour plus d'informations sur la création de classes personnalisées, consultez le Chapitre 6, « Classes », à la page 197.

Opérateur conditionnel et syntaxe de remplacement

Si vous aimez les raccourcis, vous pouvez utiliser l'opérateur conditionnel (`?:`), également appelé *expressions conditionnelles*. L'opérateur conditionnel permet de convertir des instructions `if..else` simples en une seule ligne de code. Il permet donc de réduire la quantité de code rédigé dans un même objectif, mais le rend du même coup plus difficile à lire.

La condition suivante est écrite en version longue et permet de vérifier que la variable `numTwo` est supérieure à zéro, ce qui renvoie le résultat `numOne/numTwo` ou une chaîne de `carrot` :

```
var numOne:Number = 8;
var numTwo:Number = 5;
if (numTwo > 0) {
  trace(numOne / numTwo); // 1.6
} else {
  trace("carrot");
}
```

L'utilisation d'une expression conditionnelle permet d'écrire ce même code dans le format suivant :

```
var numOne:Number = 8;
var numTwo:Number = 0;
trace((numTwo > 0) ? numOne/numTwo : "carrot");
```

Comme vous pouvez le voir, la syntaxe abrégée nuit à la lisibilité et doit donc être évitée. Si vous êtes obligé d'utiliser des opérateurs conditionnels, placez la première condition (avant le point d'interrogation [`?`]) entre parenthèses. Cette précaution améliore la lisibilité du code ActionScript. Vous trouverez ci-dessous un exemple de code ActionScript dont la lisibilité a été améliorée :

```
var numOne:Number;
(numOne >= 5) ? numOne : -numOne;
```

Vous pouvez écrire une instruction conditionnelle qui renvoie une valeur booléenne, comme dans l'exemple suivant :

```
if (cartArr.length > 0) {
  return true;
} else {
  return false;
}
```

Cependant, comparé avec le code précédent, le code ActionScript de l'exemple suivant est préférable :

```
return (cartArr.length > 0);
```

Le deuxième fragment de code est plus concis et comporte moins d'expressions à évaluer. Il est plus facile à lire et à comprendre.

Lorsque vous écrivez des conditions complexes, il est conseillé de les regrouper entre parenthèses [()]. Si vous n'utilisez pas de parenthèses, vous (ou toute autre personne travaillant sur le code ActionScript) risquez de subir des erreurs au niveau de la priorité des opérateurs. Pour plus d'informations sur la priorité des opérateurs, consultez la section « Priorité et associativité des opérateurs », à la page 149.

Par exemple, le code suivant ne met pas la condition entre parenthèses :

```
if (fruit == "apple" && veggie == "leek") {}
```

Le code suivant met les conditions entre parenthèses de manière appropriée :

```
if ((fruit == "apple") && (veggie == "leek")) {}
```

Répétition d'actions à l'aide de boucles

ActionScript peut répéter une action un certain nombre de fois ou tant qu'une condition spécifique existe. Les boucles permettent de répéter une suite d'instructions lorsqu'une condition particulière est `true` (vraie). Il existe quatre types de boucles dans ActionScript : `for`, `for..in`, `while` et `do..while`. Chaque type de boucle se comporte légèrement différemment et sert des objectifs distincts.

La plupart des boucles utilisent un compteur d'un certain type pour contrôler le nombre d'exécutions d'une boucle. Chaque exécution d'une boucle est appelée *itération*. Vous pouvez déclarer une variable et rédiger une instruction qui augmente ou diminue sa valeur à chaque exécution de la boucle. Dans l'action `for`, le compteur et l'instruction qui l'incrémente font partie de l'action.

Boucle	Description
Boucles `for`	Répéter une action en utilisant un compteur intégré.
Boucles `for..in`	Effectuer une itération sur les enfants d'un clip ou d'un objet.
Boucles `while`	Répéter une action tant qu'une condition est vraie.
Boucles `do..while`	Similaires aux boucles while, sauf que l'expression est évaluée à la fin du bloc de code et la boucle est toujours exécutée au moins une fois.

La plus répandue est la boucle `for`, qui passe sur un bloc de code un nombre de fois prédéfini. Par exemple, pour exécuter une série d'instructions sur chaque élément d'un tableau, utilisez une boucle `for` et exécutez-la de 0 jusqu'au nombre d'éléments du tableau. La boucle `for..in` est également très utile lorsque vous souhaitez faire une boucle sur chaque paire nom/valeur d'un objet, puis exécuter un type d'action spécifique. Cela peut se révéler très pratique lors du débogage de projets Flash si vous devez afficher les valeurs chargées à partir de sources externes, telles que des services Web ou des fichiers texte/XML externes. Les deux derniers types de boucles (`while` et `do..while`) sont très pratiques lorsqu'il s'agit de passer en boucle sur une série d'instructions, sans nécessairement connaître le nombre de passages nécessaires. Dans ce cas, utilisez une boucle `while` qui tourne tant qu'une certaine condition est vraie.

ActionScript peut répéter une action un certain nombre de fois ou tant qu'une condition spécifique existe. Utilisez les actions `while`, `do..while`, `for` et `for..in` pour créer des boucles. Cette section présente des informations générales sur ces boucles. Pour plus d'informations sur chacune de ces boucles, consultez les procédures suivantes.

Pour répéter une action tant qu'une condition existe :

- Utilisez l'instruction `while`.

  La boucle `while` évalue une expression et exécute le code dans le corps de la boucle si l'expression est `true`. L'expression est évaluée à nouveau après l'exécution de chaque instruction du corps. Dans l'exemple suivant, la boucle est exécutée à quatre reprises :

  ```
  var i:Number = 4;
  while (i > 0) {
    myClip.duplicateMovieClip("newMC" + i, i, {_x:i*20, _y:i*20});
    i--;
  }
  ```

  Vous pouvez utiliser l'instruction `do..while` pour créer le même genre de boucle qu'avec une boucle `while`. Dans une boucle `do..while`, l'expression est évaluée à la fin du bloc de code et la boucle est toujours exécutée au moins une fois.

  Ceci est illustré par l'exemple suivant :

  ```
  var i:Number = 4;
  do {
    myClip.duplicateMovieClip("newMC" + i, i, {_x:i*20, _y:i*20});
    i--;
  } while (i > 0);
  ```

  Pour plus d'informations sur l'instruction `while`, consultez la section « Utilisation des boucles while », à la page 129.

Pour répéter une action en utilisant un compteur intégré :

- Utilisez l'instruction `for`.

  La plupart des boucles utilisent un compteur d'un certain type pour contrôler le nombre d'exécutions d'une boucle. Chaque exécution d'une boucle est appelée *itération*.
  Vous pouvez déclarer une variable et rédiger une instruction qui augmente ou diminue sa valeur à chaque exécution de la boucle. Dans l'action `for`, le compteur et l'instruction qui l'incrémente font partie de l'action.

  Dans l'exemple suivant, la première expression (`var i:Number = 4`) est l'expression initiale qui est évaluée avant la première itération. La deuxième expression (`i > 0`) est la condition contrôlée avant chaque exécution de la boucle. La troisième expression (`i--`) est appelée *post-expression* et est évaluée après chaque exécution de la boucle.

  ```
  for (var i:Number = 4; i > 0; i--) {
    myClip.duplicateMovieClip("newMC" + i, i, {_x:i*20, _y:i*20});
  }
  ```

  Pour plus d'informations sur l'instruction `for`, consultez la section « Utilisation des boucles for », à la page 126.

Pour passer en boucle sur les enfants d'un clip ou d'un objet :

- Utilisez l'instruction `for..in`.

  Les enfants sont composés d'autres clips, fonctions, objets et variables. L'exemple suivant utilise l'instruction `trace` pour envoyer les résultats dans le panneau de sortie :

  ```
  var myObject:Object = {name:'Joe', age:25, city:'San Francisco'};
  var propertyName:String;
  for (propertyName in myObject) {
    trace("myObject has the property: " + propertyName + ", with the
    value: " + myObject[propertyName]);
  }
  ```

  Cet exemple donne les résultats suivants dans le panneau de sortie :

  ```
  myObject has the property: name, with the value: Joe
  myObject has the property: age, with the value: 25
  myObject has the property: city, with the value: San Francisco
  ```

  Vous pouvez souhaiter que votre script boucle sur un type particulier d'enfants, par exemple, seulement sur les enfants d'un clip. Pour ce faire, utilisez `for..in` en conjonction avec l'opérateur `typeof`. Dans l'exemple suivant, une occurrence de clip enfant (appelée `occurrence2`) est située dans une occurrence de clip sur la scène. Ajoutez le code ActionScript suivant à l'Image 1 du scénario :

  ```
  for (var myName in this) {
    if (typeof (this[myName]) == "movieclip") {
      trace("I have a movie clip child named " + myName);
    }
  }
  ```

Pour plus d'informations sur l'instruction `for..in`, consultez la section « Utilisation des boucles for..in », à la page 127.

AVERTISSEMENT	Dans Flash, les itérations s'exécutent très rapidement dans Flash Player, mais les boucles dépendent essentiellement du processeur. La quantité de ressources consommées par le processeur évolue en fonction du nombre d'itérations de la boucle et du nombre d'instructions exécutées dans chaque bloc. Les boucles mal écrites peuvent générer des problèmes de performance et de stabilité.

Pour plus d'informations sur chaque instruction, consultez les sections suivantes de ce chapitre, telles que « Utilisation des boucles while », à la page 129, et leurs entrées respectives dans le *Guide de référence du langage ActionScrip 2.0*.

Création et terminaison des boucles

L'exemple suivant présente un simple tableau de noms de mois. Une boucle `for` itère de `0` au nombre d'éléments présents dans le tableau et affiche chaque élément dans le panneau de sortie.

```
var monthArr:Array = new Array("Jan", "Feb", "Mar", "Apr", "May", "Jun",
  "Jul", "Aug", "Sep", "Oct", "Nov", "Dec");
var i:Number;
for (i = 0; i < monthArr.length; i++) {
  trace(monthArr[i]);
}
```

Lorsque vous manipulez des tableaux, simples ou complexes, vous devez connaître la condition appelée *boucle sans fin*. Comme son nom l'indique, il s'agit d'une boucle qui n'a aucune condition de fin. Cela entraîne un véritable problème (le blocage de votre application Flash) : votre document Flash ne répond plus dans le navigateur Web ou a un comportement incohérent. Le code suivant est un exemple de boucle sans fin :

```
// MAUVAIS CODE- crée une boucle sans fin
// A vos risques et périls !
var i:Number;
for (i = 0; i < 10; i--) {
  trace(i);
}
```

La valeur de `i` est initialisée sur 0 et la condition de fin survient lorsque `i` est supérieur ou égal à 10, la valeur de `i` étant décrémentée après chaque itération. L'erreur vous saute probablement déjà aux yeux : si la valeur de `i` diminue après chaque itération, la condition de fin n'arrive jamais. Le résultat varie selon les ordinateurs, et la rapidité du blocage dépend de la cadence du processeur et d'autres facteurs. Par exemple, la boucle s'exécute environ 142 620 fois avant d'afficher un message d'erreur sur un ordinateur donné.

Le message d'erreur suivant s'affiche dans une boîte de dialogue :

```
A script in this movie is causing Flash Player to run slowly. If it
  continues to run, your computer may become unresponsive. Do you want to
  abort the script?
```

Lorsque vous utilisez des boucles (et en particulier `while` et `do..while`), n'oubliez jamais de vérifier que la boucle peut se terminer correctement, et ne continue pas sans fin.

Pour plus d'informations sur le contrôle des boucles, consultez la section « Utilisation de l'instruction switch », à la page 116.

Utilisation des boucles for

La boucle `for` permet de faire une itération sur une variable dans une plage de valeurs spécifiques. La boucle `for` est très utile lorsque vous savez exactement combien de fois une série d'instructions ActionScript doit être répétée. Elle peut se révéler très utile lorsque vous souhaitez dupliquer un clip de la scène un certain nombre de fois ou passer en boucle dans un tableau et exécuter une tâche sur chacun de ses éléments. La boucle `for` répète une action à l'aide d'un compteur intégré. Le compteur et l'instruction qui l'incrémente font tous deux partie de l'instruction `for`. Rédigez les instructions `for` au format suivant :

```
for (init; condition; update) {
  // instructions ;
}
```

Vous devez fournir trois expressions dans une instruction `for` : une variable définie sur une valeur initiale, une instruction conditionnelle qui détermine le moment où la boucle prend fin et une expression qui modifie la valeur de la variable à chaque boucle. Par exemple, le code suivant boucle à cinq reprises. La valeur de la variable `i` commence à `0` et termine à `4`. La sortie indique les nombres `0` à `4`, chacun sur sa propre ligne.

```
var i:Number;
for (i = 0; i < 5; i++) {
  trace(i);
}
```

Dans l'exemple suivant, la première expression (`i = 0`) est l'expression initiale évaluée avant la première itération. La deuxième expression (`i < 5`) est la condition contrôlée avant chaque exécution de la boucle. La troisième expression (`i++`) est appelée *post-expression* et est évaluée après chaque exécution de la boucle.

Pour créer une boucle for :

1. Choisissez Fichier > Nouveau, puis Document Flash.
2. Créez un clip sur la scène.
3. Dans le panneau Bibliothèque, cliquez du bouton droit sur le symbole du clip et choisissez Liaison dans le menu contextuel.
4. Cochez la case Exporter pour ActionScript, puis tapez **libraryLinkageClassName** dans le champ Classe. Cliquez sur OK.
5. Sélectionnez l'image 1 du scénario et saisissez le code ActionScript suivant dans le panneau Actions :

```
var i:Number;
for (i = 0; i < 5; i++) {
  this.attachMovie("libraryLinkageClassName", "clip" + i + "_mc", i,
  {_x:(i * 100)});
}
```

6. Choisissez Contrôle > Tester l'animation pour tester le code dans Flash Player.

   Remarquez la duplication des cinq clips en haut de la scène. Ce code ActionScript reproduit le symbole de clip de la bibliothèque et repositionne les clips sur la scène, aux coordonnées *x* de 0, 100, 200, 300 et 400 pixels. La boucle s'exécute à cinq reprises, une valeur comprise entre 0 et 4 étant affectée à la variable `i`. A la dernière itération, la valeur de `i` devient 4 et la seconde expression (`i < 5`) n'étant plus vraie, la boucle s'arrête.

Vous devez inclure un espace après chaque expression d'une instruction `for`. Pour plus d'informations, reportez-vous à `for statement` dans le *Guide de référence du langage ActionScript 2.0*.

Utilisation des boucles for..in

L'instruction `for..in` permet d'exécuter une boucle (ou de faire une *itération*) sur les enfants d'un clip, les propriétés d'un objet ou les éléments d'un tableau. Les enfants, référencés précédemment, sont composés d'autres clips, fonctions, objets et variables. La boucle `for..in` est souvent utilisée pour passer en boucle sur les occurrences d'un scénario ou sur les paires clé/valeur d'un objet. Ce passage en boucle sur des objets se révèle efficace pour déboguer les applications car il permet de voir les données renvoyées par les services Web ou les documents externes, tels que les fichiers texte ou XML.

Par exemple, utilisez une boucle `for...in` pour faire une itération sur les propriétés d'un objet générique (les propriétés d'un objet n'étant pas conservées dans un ordre particulier, elles apparaissent dans un ordre imprévisible) :

```
var myObj:Object = {x:20, y:30};
for (var i:String in myObj) {
  trace(i + ": " + myObj[i]);
}
```

Ce code donne les résultats suivants dans le panneau de sortie :

```
x: 20
y: 30
```

Vous pouvez également faire une itération sur les éléments d'un tableau :

```
var myArray:Array = ["one", "two", "three"];
for (var i:String in myArray) {
  trace(myArray[i]);
}
```

Ce code donne les résultats suivants dans le panneau de sortie :

```
three
two
one
```

Pour plus d'informations sur les objets et les propriétés, consultez la section « Type de données Object », à la page 43.

REMARQUE

Vous ne pouvez pas itérer sur les propriétés d'un objet lorsqu'il s'agit d'une occurrence de classe personnalisée, sauf si la classe est dynamique. Et même dans ce dernier cas, vous ne pouvez faire d'itération que sur les propriétés ajoutées dynamiquement.

REMARQUE

Les accolades (`{}`), qui servent normalement à entourer le bloc d'instructions que la boucle `for..in` doit exécuter, peuvent être omises si une seule instruction s'exécute.

L'exemple suivant utilise une boucle `for..in` sur les propriétés d'un objet :

Pour créer une boucle for :

1. Choisissez Fichier > Nouveau, puis Document Flash.
2. Sélectionnez l'image 1 du scénario et saisissez le code ActionScript suivant dans le panneau Actions :

```
var myObj:Object = {name:"Tara", age:27, city:"San Francisco"};
var i:String;
for (i in myObj) {
  trace("myObj." + i + " = " + myObj[i]);
}
```

3. Choisissez Contrôle > Tester l'animation pour tester le code dans Flash Player.

   Lorsque vous testez le fichier SWF, le message suivant s'affiche dans le panneau de sortie :

```
myObj.name = Tara
myObj.age = 27
myObj.city = San Francisco
```

Si vous écrivez une boucle `for..in` dans un fichier de classe (fichier ActionScript externe), les membres de l'occurrence ne seront plus disponibles dans la boucle, contrairement aux membres statiques. Cependant, si vous écrivez une boucle `for..in` dans un fichier FLA pour une occurrence de la classe, les membres de l'occurrence restent disponibles, mais pas les membres statiques. Pour plus d'informations sur la rédaction de fichiers de classe, consultez le Chapitre 6, « Classes », à la page 197. Pour plus d'informations, reportez-vous à `for..in statement` dans le *Guide de référence du langage ActionScript 2.0*.

Utilisation des boucles while

L'instruction `while` permet de répéter une action tant qu'une condition existe, comme l'instruction `if` qui se répète tant que la condition est `true` (vraie).

Une boucle `while` évalue une expression et exécute le code présent dans la boucle si l'expression est `true`. Si la condition renvoie `true`, une instruction ou une série d'instructions est exécutée avant de revenir en arrière et d'évaluer de nouveau la condition. Lorsque la condition renvoie `false` (faux), l'instruction ou la série d'instructions est ignorée et la boucle se termine. Les boucles `while` se révèlent très utiles lorsque vous ne savez pas précisément combien de boucles doivent être exécutées sur un bloc de code.

Par exemple, ce code envoie les nombres suivants dans le panneau de sortie :

```
var i:Number = 0;
while (i < 5) {
  trace(i);
  i++;
}
```

Les nombres suivants doivent s'afficher dans le panneau de sortie :

```
0
1
2
3
4
```

L'un des inconvénients que présente la boucle `while` par rapport à la boucle `for` est que les risques de boucles sans fin sont plus importants avec les boucles `while`. Par exemple, le code qui utilise la boucle `for` ne passera pas la compilation si vous omettez l'expression qui incrémente la variable du compteur, alors que le code qui utilise la boucle `while` sera compilé. Et sans l'expression qui incrémente `i`, la boucle se poursuit sans fin.

Pour créer et utiliser une boucle `while` dans un fichier FLA, procédez comme dans cet exemple.

Pour créer une boucle while :

1. Choisissez Fichier > Nouveau, puis Document Flash.
2. Ouvrez le panneau Composants et faites glisser un composant DataSet sur la scène.
3. Ouvrez l'inspecteur des propriétés (Fenêtre > Propriétés > Propriétés) et entrez le nom d'occurrence **users_ds**.
4. Sélectionnez l'image 1 du scénario et saisissez le code ActionScript suivant dans le panneau Actions :

```
var users_ds:mx.data.components.DataSet;
//
users_ds.addItem({name:"Irving", age:34});
users_ds.addItem({name:"Christopher", age:48});
users_ds.addItem({name:"Walter", age:23});
//
users_ds.first();
while (users_ds.hasNext()) {
  trace("name:" + users_ds.currentItem["name"] + ", age:" +
  users_ds.currentItem["age"]);
  users_ds.next();
}
```

5. Sélectionnez Contrôle > Tester l'animation pour tester le document.

   Les informations suivantes apparaissent dans le panneau de sortie :

```
name:Irving, age:34
name:Christopher, age:48
name:Walter, age:23
```

Pour plus d'informations, reportez-vous à l'`instruction while` dans le `Guide de référence du langage ActionScript 2.0`.

Présentation des boucles do..while

Vous pouvez utiliser l'instruction `do..while` pour créer le même genre de boucle qu'avec une boucle `while`. Toutefois, dans une boucle `do..while`, l'expression est évaluée à la fin du bloc de code (après son exécution) et la boucle est toujours exécutée au moins une fois.
Les instructions ne sont exécutées que lorsque la condition est `true` (vraie).

Le code suivant présente un exemple simple de boucle `do..while` générant une sortie même lorsque la condition n'est pas remplie.

```
var i:Number = 5;
do {
  trace(i);
  i++;
} while (i < 5);
// Résultat : 5
```

Lorsque vous utilisez des boucles, faites attention à ne pas créer de boucle sans fin. Si la condition d'une boucle `do..while` renvoie `true` (vrai) en permanence, une boucle sans fin est créée et entraîne l'affichage d'un avertissement ou le blocage de Flash Player. Si vous connaissez le nombre d'itérations nécessaires, choisissez plutôt une boucle `for`. Pour plus d'informations et des exemples de l'`instruction do..while`, consultez le *Guide de référence du langage ActionScript 2.0*.

Utilisation de boucles imbriquées dans le code ActionScript

L'exemple suivant décrit la construction d'un tableau d'objets et l'affichage de chaque valeur dans la structure imbriquée. Cet exemple présente l'utilisation de la boucle `for` pour effectuer une itération sur chaque élément du tableau et de la boucle `for..in` pour effectuer une itération sur chaque paire clé/valeur dans les objets imbriqués.

Pour imbriquer une boucle dans une autre :

1. Créez un document Flash.
2. Choisissez Fichier > Enregistrer sous et nommez le document **loops.fla**.
3. Ajoutez le code suivant à l'image 1 du scénario :

```
var myArr:Array = new Array();
myArr[0] = {name:"One", value:1};
myArr[1] = {name:"Two", value:2};
//
var i:Number;
var item:String;
for (i = 0; i < myArr.length; i++) {
  trace(i);
  for (item in myArr[i]) {
    trace(item + ": " + myArr[i][item]);
  }
  trace("");
}
```

4. Choisissez Contrôle > Tester l'animation pour tester votre code.

   Les informations suivantes apparaissent dans le panneau de sortie.

```
0
name: One
value: 1

1
name: Two
value: 2
```

   Comme vous connaissez le nombre d'éléments présents dans le tableau, vous pouvez utiliser une simple boucle `for` pour passer sur chaque élément. Chaque objet du tableau pouvant présenter des paires nom/valeur différentes, vous pouvez utiliser une boucle `for..in` pour effectuer une itération sur chaque valeur et afficher les résultats dans le panneau de sortie.

Présentation des tableaux

Un *tableau* est un objet dont les propriétés sont identifiées par des nombres représentant leurs positions dans la structure. Avant tout, un tableau est une liste d'éléments. Il est important de ne pas oublier que tous les éléments d'un tableau ne sont pas obligatoirement du même type de données. Vous pouvez combiner des nombres, des dates, des chaînes, des objets ou même ajouter un tableau imbriqué dans chaque index de tableau.

L'exemple suivant présente un simple tableau de noms de mois.

```
var myArr:Array = new Array();
myArr[0] = "January";
myArr[1] = "February";
myArr[2] = "March";
myArr[3] = "April";
```

Le tableau précédent de noms de mois peut également être réécrit comme suit :

```
var myArr:Array = new Array("January", "February", "March", "April");
```

Vous pouvez également utiliser une syntaxe abrégée :

```
var myArr:Array = ["January", "February", "March", "April"];
```

Un tableau correspond à une structure de données. On peut le comparer à un bâtiment au sein duquel chaque étage contiendrait un morceau de données différent (tel que la *comptabilité* au niveau 3 et *l'ingénierie* au niveau 5). Ainsi, il est possible de stocker des types de données différents, y compris d'autres tableaux, dans un même tableau. Chaque niveau du bâtiment peut contenir de nombreux types de contenus (la *direction* et la *comptabilité* peuvent partager le niveau 3).

Le tableau contient des *éléments*, qui correspondent à chaque étage du bâtiment. Chaque élément occupe une position numérique (l'*index*), à laquelle vous faites référence dans le tableau. De la même façon, chaque niveau du bâtiment est associé à un numéro d'étage. Chaque élément peut contenir une part de données (un nombre, une chaîne, une valeur booléenne ou même un tableau ou un objet) ou être vide.

Vous pouvez également contrôler et modifier le tableau lui-même. Par exemple, vous pouvez déménager le service d'ingénierie au rez-de-chaussée du bâtiment. Les tableaux vous permettent de déplacer les valeurs qu'ils contiennent et de modifier leur taille (disons de rénover le bâtiment et d'ajouter ou de supprimer des étages). Ainsi, vous pouvez ajouter ou supprimer des éléments et déplacer les valeurs vers d'autres éléments.

De ce fait, le bâtiment (tableau) contient des étages (éléments) numérotés (index), et chaque étage contient un ou plusieurs services (valeurs).

Pour plus d'informations sur la modification des tableaux, consultez la section « Modification des tableaux », à la page 135. Pour plus d'informations sur l'utilisation des tableaux et des index, consultez la section « Utilisation des tableaux », à la page 133. Pour plus d'informations sur l'ajout ou la suppression d'éléments, consultez la section « Ajout et suppression d'éléments », à la page 137. Pour plus d'informations sur l'opérateur d'accès au tableau, consultez la section « Utilisation des opérateurs point et d'accès au tableau », à la page 153.

Pour un exemple de fichier source array.fla qui décrit la manipulation des tableaux à l'aide du code ActionScript, consultez la page d'exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Arrays afin d'accéder à l'exemple. Le code de cet exemple crée un tableau et trie, ajoute et supprime des éléments de deux composants List.

Utilisation des tableaux

Vous pouvez utiliser des tableaux de différentes manières dans vos projets. Vous pouvez y stocker des listes d'objets, par exemple un lot d'articles renvoyés. Si vous chargez les données à partir de serveurs Web distants, vous pouvez même les recevoir sous forme de tableaux d'objets imbriqués. Les tableaux contiennent souvent des données de format similaire. Par exemple, si vous développez une application audio dans Flash, vous pouvez stocker la liste de lecture de l'utilisateur sous forme de tableau d'informations sur les chansons, stockées dans des objets. Chaque objet contient le titre du morceau, le nom de l'interprète, la durée de la chanson, l'emplacement du ficher audio (par exemple MP3) ou toute autre information devant être associée à un fichier particulier.

L'emplacement d'un élément au sein du tableau est appelé *index*. Tous les tableaux sont basés sur zéro, ce qui signifie que le premier élément du tableau est [0], le deuxième est [1], etc.

Il existe différents types de tableaux, comme vous le découvrirez dans les sections suivantes. Les tableaux les plus courants utilisent un index numérique pour rechercher un élément particulier dans un *tableau indexé*. Le second type de tableau, appelé *tableau associatif*, recherche les informations à l'aide d'un index de texte, non numérique. Pour plus d'informations sur les tableaux les plus courants, consultez la section « Présentation des tableaux », à la page 132. Pour plus d'informations sur les tableaux associatifs, consultez la section « Création de tableaux associatifs », à la page 141. Pour plus d'informations sur les tableaux multidimensionnels, consultez la section « Création de tableaux multidimensionnels », à la page 138. Pour plus d'informations sur l'opérateur d'accès au tableau, consultez la section « Utilisation des opérateurs point et d'accès au tableau », à la page 153.

La classe intégrée Array vous permet d'accéder aux tableaux et de les manipuler. Pour créer un objet Array, utilisez le constructeur `new Array()` ou l'opérateur d'accès au tableau (`[]`). Pour accéder aux éléments d'un tableau, utilisez également l'opérateur d'accès (`[]`). L'exemple suivant utilise un tableau indexé.

Pour utiliser des tableaux dans votre code :

1. Créez un document Flash, puis enregistrez-le sous le nom **basicArrays.fla**.
2. Ajoutez le code ActionScript suivant à l'Image 1 du scénario :

```
// définition d'un nouveau tableau
var myArr:Array = new Array();
// définition des valeurs de deux index
myArr[1] = "value1";
myArr[0] = "value0";
// itération sur les éléments du tableau
var i:String;
for (i in myArr) {
  // suivi des paires clé/valeur
  trace("key: " + i + ", value: " + myArr[i]);
}
```

   Dans la première ligne de code ActionScript, vous définissez un nouveau tableau pour stocker les valeurs. Vous définissez ensuite les données (`value0` et `value1`) sur deux index du tableau. Vous utilisez une boucle `for..in` pour faire une itération sur chaque élément de ce tableau et afficher les paires clé/valeur dans le panneau de sortie à l'aide d'une instruction trace.
3. Choisissez Contrôle > Tester l'animation pour tester votre code.

   Le texte suivant apparaît dans le panneau de sortie :

```
key: 0, value: value0
key: 1, value: value1
```

   Pour plus d'informations sur les boucles `for..in`, consultez la section « Utilisation des boucles for..in », à la page 127.

Pour plus d'informations sur la création de types de tableaux différents, consultez les sections suivantes :

- « Création de tableaux indexés », à la page 137
- « Création de tableaux multidimensionnels », à la page 138
- « Création de tableaux associatifs », à la page 141

Pour un exemple de fichier source array.fla qui décrit la manipulation des tableaux à l'aide du code ActionScript, consultez la page d'exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Arrays afin d'accéder à l'exemple. Le code de cet exemple crée un tableau et trie, ajoute et supprime des éléments de deux composants List.

Modification des tableaux

Vous pouvez également contrôler et modifier un tableau avec ActionScript. Vous pouvez déplacer des valeurs dans le tableau ou modifier la taille de celui-ci. Par exemple, pour échanger les données de deux index d'un tableau, vous pouvez utiliser le code suivant :

```
var buildingArr:Array = new Array();
buildingArr[2] = "Accounting";
buildingArr[4] = "Engineering";
trace(buildingArr); // undefined,undefined,Accounting,undefined,Engineering

var temp_item:String = buildingArr[2];
buildingArr[2] = buildingArr[4];
buildingArr[4] = temp_item;
trace(buildingArr); // undefined,undefined,Engineering,undefined,Accounting
```

Vous vous demandez peut-être pourquoi vous avez dû créer une variable temporaire dans l'exemple précédent. En fait, la copie du contenu de l'index 4 du tableau dans l'index 2 et vice versa, aurait entraîné la perte du contenu original de l'index 2 du tableau. En copiant la valeur de l'un des index dans une variable temporaire, vous enregistrez la valeur et la recopiez ensuite dans votre code. Par exemple, si vous utilisez le code suivant, vous pouvez voir que la valeur de l'index 2 du tableau (Accounting) a été perdue. Vous avez maintenant deux équipes d'ingénieurs mais plus de comptables.

```
// méthode incorrecte (pas de variable temporaire)
buildingArr[2] = buildingArr[4];
buildingArr[4] = buildingArr[2];
trace(buildingArr); //
  undefined,undefined,Engineering,undefined,Engineering
```

Pour un exemple de fichier source array.fla qui décrit la manipulation des tableaux à l'aide du code ActionScript, consultez la page d'exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Arrays afin d'accéder à l'exemple. Le code de cet exemple crée un tableau et trie, ajoute et supprime des éléments de deux composants List.

Référencement et recherche de longueur

Pour manipuler les tableaux, il est souvent nécessaire de connaître le nombre d'éléments qu'ils contiennent. Cette information se révèle très utile pour l'écriture des boucles `for` chargées d'effectuer une itération sur chaque élément du tableau et d'exécuter une série d'instructions. Le code suivant en est un exemple :

```
var monthArr:Array = new Array("Jan", "Feb", "Mar", "Apr", "May", "Jun",
  "Jul", "Aug", "Sep", "Oct", "Nov", "Dec");
trace(monthArr); // Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec
trace(monthArr.length); // 12
var i:Number;
for (i = 0; i < monthArr.length; i++) {
  monthArr[i] = monthArr[i].toUpperCase();
}
trace(monthArr); // JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
```

Dans l'exemple précédent, vous créez un tableau et le remplissez avec les noms de mois. Le contenu est affiché, de même que la longueur du tableau. Une boucle `for` fait une itération sur chaque élément du tableau, met la valeur en majuscule, et le contenu du tableau est affiché de nouveau.

Dans le code ActionScript suivant, si vous créez un élément à l'index 5 d'un tableau, la longueur du tableau renvoie `6` (le tableau étant basé sur zéro), et non le nombre réel d'éléments du tableau auquel vous vous attendiez :

```
var myArr:Array = new Array();
myArr[5] = "five";
trace(myArr.length); // 6
trace(myArr); // undefined,undefined,undefined,undefined,undefined,five
```

Pour plus d'informations sur les boucles `for`, consultez la section « Utilisation des boucles for », à la page 126. Pour plus d'informations sur l'opérateur d'accès au tableau, consultez la section « Utilisation des opérateurs point et d'accès au tableau », à la page 153.

Pour un exemple de fichier source array.fla qui décrit la manipulation des tableaux à l'aide du code ActionScript, consultez la page d'exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Arrays afin d'accéder à l'exemple. Le code de cet exemple crée un tableau et trie, ajoute et supprime des éléments de deux composants List.

Ajout et suppression d'éléments

Chaque élément d'un tableau occupe une position numérique (l'index), à laquelle vous faites référence. Chaque élément peut contenir des données ou être vide. Un élément peut contenir les données suivantes : un nombre, une chaîne, une valeur booléenne ou encore un tableau ou un objet.

Lorsque vous créez des éléments dans un tableau, il est préférable, dans la mesure du possible, de créer les index en séquence. Le débogage de vos applications s'en trouvera simplifié. A la section « Référencement et recherche de longueur », à la page 136, vous avez vu que si vous affectez une seule valeur à l'index 5 d'un tableau, la longueur du tableau renvoie 6. Cinq valeurs indéfinies sont alors insérées dans le tableau.

L'exemple suivant décrit la création d'un nouveau tableau, la suppression d'un élément au niveau d'un index particulier et l'ajout et le remplacement des données d'un index du tableau :

```
var monthArr:Array = new Array("Jan", "Feb", "Mar", "Apr", "May", "Jun",
  "Jul", "Aug", "Sep", "Oct", "Nov", "Dec");
delete monthArr[5];
trace(monthArr); // Jan,Feb,Mar,Apr,May,undefined,Jul,Aug,Sep,Oct,Nov,Dec
trace(monthArr.length); // 12
monthArr[5] = "JUN";
trace(monthArr); // Jan,Feb,Mar,Apr,May,JUN,Jul,Aug,Sep,Oct,Nov,Dec
```

Bien que vous ayez supprimé l'élément à l'index 5, la longueur du tableau demeure 12 car l'élément de l'index 5 n'a pas entièrement disparu mais a été remplacé par une chaîne vide.

Pour un exemple de fichier source array.fla qui décrit la manipulation des tableaux à l'aide du code ActionScript, consultez la page d'exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Arrays afin d'accéder à l'exemple. Le code de cet exemple crée un tableau et trie, ajoute et supprime des éléments de deux composants List.

Création de tableaux indexés

Les tableaux indexés stockent des suites d'une ou plusieurs valeurs. Vous pouvez rechercher les éléments par leur position dans le tableau, ce que vous avez peut-être fait dans les sections précédentes. Le premier index correspond toujours au nombre `0`. L'index est ensuite incrémenté d'une unité pour chaque élément ajouté consécutivement au tableau. Vous pouvez créer un tableau indexé en appelant le constructeur de classe Array ou en initialisant le tableau avec un littéral de tableau. L'exemple suivant permet de créer des tableaux à l'aide du constructeur Array et d'un littéral de tableau.

Pour créer un tableau indexé :

1. Créez un document Flash, puis enregistrez-le sous le nom **indexArray.fla**.
2. Ajoutez le code ActionScript suivant à l'Image 1 du scénario :

```
var myArray:Array = new Array();
myArray.push("one");
myArray.push("two");
myArray.push("three");
trace(myArray); // one,two,three
```

   Dans la première ligne de code ActionScript, vous définissez un nouveau tableau pour stocker les valeurs.
3. Choisissez Contrôle > Tester l'animation pour tester votre code.

   Le texte suivant apparaît dans le panneau de sortie :

```
one,two,three
```
4. Revenez dans l'outil de programmation, puis supprimez le code dans le panneau Actions.
5. Ajoutez le code ActionScript suivant à l'Image 1 du scénario :

```
var myArray:Array = ["one", "two", "three"];
trace(myArray); // one,two,three
```

   Dans ce code, vous utilisez le littéral de tableau pour définir un nouveau tableau pour votre code. Ce code est similaire à celui que vous avez écrit à l'étape 2. Lorsque vous le testez, le même résultat apparaît dans le panneau de sortie.

Création de tableaux multidimensionnels

ActionScript permet d'implémenter des *tableaux imbriqués* correspondant avant tout à des tableaux de tableaux. Les tableaux imbriqués, également appelés *tableaux multidimensionnels*, sont considérés comme des matrices ou des grilles. Lorsque vous programmez, vous pouvez utiliser les tableaux multidimensionnels pour reproduire ce type de structures. Par exemple, un échiquier est une grille de huit colonnes et huit lignes qui peut être reproduit sous forme de tableau contenant huit éléments, chacun d'eux correspondant également à un tableau contenant huit éléments.

Prenons par exemple une liste de tâches stockée sous forme de tableau indexé de chaînes :

```
var tasks:Array = ["wash dishes", "take out trash"];
```

Pour stocker une liste de tâches distincte pour chaque jour de la semaine, vous pouvez créer un tableau multidimensionnel avec un élément pour chaque jour. Chaque élément contient à son tour un tableau indexé qui stocke la liste des tâches.

ATTENTION	Lorsque vous utilisez l'opérateur d'accès au tableau, le compilateur ActionScript ne peut pas vérifier si l'élément auquel vous avez accédé est une propriété valable de l'objet.

Pour créer un tableau multidimensionnel de base et récupérer les éléments du tableau :

1. Créez un document Flash, puis enregistrez-le sous le nom **multiArray1.fla**.
2. Ajoutez le code ActionScript suivant à l'Image 1 du scénario :

```
var twoDArray:Array = new Array(new Array("one","two"), new
  Array("three", "four"));
trace(twoDArray);
```

   Ce tableau, `twoDArray`, se compose de deux éléments. Ces éléments sont eux-mêmes des tableaux de deux éléments. Dans ce cas, `twoDArray` est le tableau principal qui contient deux tableaux imbriqués.
3. Choisissez Contrôle > Tester l'animation pour tester le code. Les informations suivantes apparaissent dans le panneau de sortie :

```
one,two,three,four
```

4. Reprenez l'outil de programmation et ouvrez le panneau Actions. Commentez l'instruction `trace`, comme suit :

```
// trace(twoDArray);
```

5. Ajoutez le code ActionScript suivant à la suite de votre code dans l'image 1 du scénario :

```
trace(twoDArray[0][0]); // un
trace(twoDArray[1][1]); // quatre
```

   Pour récupérer les éléments d'un tableau multidimensionnel, utilisez plusieurs opérateurs d'accès (`[]`) après le nom du tableau de niveau supérieur. Le premier opérateur `[]` fait référence à l'index du tableau de niveau supérieur. Les opérateurs d'accès au tableau suivants font référence aux éléments des tableaux imbriqués.
6. Choisissez Contrôle > Tester l'animation pour tester le code. Les informations suivantes apparaissent dans le panneau de sortie :

```
one
four
```

Vous pouvez utiliser des boucles `for` imbriquées pour créer des tableaux multidimensionnels. L'exemple suivant décrit cette procédure.

Pour créer un tableau multidimensionnel à l'aide d'une boucle for :

1. Créez un document Flash, puis enregistrez-le sous le nom **multiArray2.fla**.
2. Ajoutez le code ActionScript suivant à l'Image 1 du scénario :

```
var gridSize:Number = 3;
var mainArr:Array = new Array(gridSize);
var i:Number;
var j:Number;
for (i = 0; i < gridSize; i++) {
  mainArr[i] = new Array(gridSize);
  for (j = 0; j < gridSize; j++) {
    mainArr[i][j] = "[" + i + "][" + j + "]";
  }
}
trace(mainArr);
```

   Ce code ActionScript crée un tableau 3 x 3 et définit la valeur de chaque noeud sur son index. Vous suivez ensuite le tableau (`mainArr`).

3. Choisissez Contrôle > Tester l'animation pour tester le code.

   Les informations suivantes apparaissent dans le panneau de sortie :

```
[0][0],[0][1],[0][2],[1][0],[1][1],[1][2],[2][0],[2][1],[2][2]
```

Vous pouvez également utiliser des boucles `for` imbriquées pour effectuer une itération sur les éléments d'un tableau multidimensionnel, comme l'illustre l'exemple suivant.

Pour utiliser une boucle for pour effectuer une itération sur un tableau multidimensionnel :

1. Créez un document Flash, puis enregistrez-le sous le nom **multiArray3.fla**.
2. Ajoutez le code ActionScript suivant à l'Image 1 du scénario :

```
// De l'exemple précédent
var gridSize:Number = 3;
var mainArr:Array = new Array(gridSize);
var i:Number;
var j:Number;
for (i = 0; i < gridSize; i++) {
  mainArr[i] = new Array(gridSize);
  for (j = 0; j < gridSize; j++) {
    mainArr[i][j] = "[" + i + "][" + j + "]";
  }
}
```

   Dans ce code, issu de l'exemple précédent, la boucle extérieure effectue une itération sur chaque élément de `mainArray`. La boucle intérieure effectue une itération sur chaque tableau imbriqué et renvoie chaque nœud du tableau.

3. Ajoutez le code ActionScript suivant dans l'image 1 du scénario, à la suite du code saisi à l'étape 2 :

```
// Itération sur les éléments
var outerArrayLength:Number = mainArr.length;
for (i = 0; i < outerArrayLength; i++) {
  var innerArrayLength:Number = mainArr[i].length;
  for (j = 0; j < innerArrayLength; j++) {
    trace(mainArr[i][j]);
  }
}
```

   Ce code ActionScript effectue une itération sur les éléments du tableau. Vous utilisez la propriété `length` de chaque tableau comme condition de la boucle.

4. Choisissez Contrôle > Tester l'animation pour voir les éléments qui apparaissent dans le panneau de sortie. Les informations suivantes apparaissent dans le panneau de sortie :

```
[0][0]
[0][1]
[0][2]
[1][0]
[1][1]
[1][2]
[2][0]
[2][1]
[2][2]
```

Pour plus d'informations sur l'utilisation des tableaux, consultez la section « Utilisation des tableaux », à la page 133. Pour plus d'informations sur les éléments de tableaux, consultez la section « Ajout et suppression d'éléments », à la page 137. Pour plus d'informations sur l'opérateur d'accès au tableau, consultez la section « Utilisation des opérateurs point et d'accès au tableau », à la page 153.

Création de tableaux associatifs

Un tableau associatif, similaire à un objet, se compose de *clés* et de *valeurs* non triées. Pour organiser les valeurs stockées, les tableaux associatifs utilisent des clés à la place des index numériques. Chaque clé est une chaîne unique et n'est associée et utilisée que pour accéder à une seule valeur. Le type des données de cette valeur peut être Number, Array, Object, etc. Lorsque vous créez du code pour rechercher la valeur associée à une clé, vous indexez ou effectuez une *recherche*. Vous utiliserez probablement les tableaux associatifs dans cet objectif.

L'association entre une clé et une valeur est généralement appelée *liaison*. La clé et la valeur sont *mises en correspondance*. Par exemple, un carnet d'adresses peut être considéré comme un tableau associatif, les noms étant les clés et les adresses de messagerie les valeurs.

REMARQUE	Les tableaux associatifs sont des collections non triées de paires de clés et de valeurs. Votre code ne doit pas s'attendre à ce que les clés d'un tel tableau se présentent dans un ordre précis.

Lorsque vous utilisez des tableaux associatifs, vous pouvez appeler l'élément désiré à l'aide d'une chaîne plutôt que d'un nombre, souvent plus simple à mémoriser. L'inconvénient est que ces tableaux ne sont pas très utiles dans une boucle car ils n'utilisent pas de nombre comme valeur d'index. Ils *sont* cependant très utiles lorsque vous avez fréquemment besoin de faire des recherches par valeurs de clés. Par exemple, dans le cas d'un tableau de noms et d'âges très sollicité, vous pouvez utiliser un tableau associatif.

L'exemple suivant décrit la création d'un objet et la définition d'une série de propriétés dans un tableau associatif.

Pour créer un tableau associatif simple :

1. Créez un document Flash.
2. Saisissez le code ActionScript suivant sur l'image 1 du scénario :

```
// Définition de l'objet à utiliser comme tableau associatif.
var someObj:Object = new Object();
// Définition d'une série de propriétés.
someObj.myShape = "Rectangle";
someObj.myW = 480;
someObj.myH = 360;
someObj.myX = 100;
someObj.myY = 200;
someObj.myAlpha = 72;
someObj.myColor = 0xDFDFDF;
// Affichage d'une propriété à l'aide d'un opérateur point et de
// la syntaxe d'accès au tableau.
trace(someObj.myAlpha); // 72
trace(someObj["myAlpha"]); // 72
```

La première ligne de code ActionScript définit un nouvel objet (`someObj`) utilisé comme tableau associatif. Ensuite, vous définissez une série de propriétés dans `someObj`. Pour finir, vous affichez une propriété sélectionnée à l'aide d'un opérateur point et de la syntaxe d'accès au tableau.

REMARQUE	Deux méthodes permettent d'accéder aux variables d'un tableau associatif : la syntaxe à point (`someObj.myColor`) et la syntaxe de tableau (`someObj['myColor']`).

3. Choisissez Contrôle > Tester l'animation pour tester votre code ActionScript.

   Le panneau de sortie affiche le nombre `72` à deux reprises, représentant les deux niveaux alpha que vous avez suivis.

Deux méthodes permettent de créer des tableaux associatifs dans ActionScript 2.0 :

- Utiliser un constructeur Object
- Utiliser un constructeur Array

Les deux méthodes sont présentées dans les exemples suivants.

REMARQUE	L'exemple précédent utilisait un constructeur Object pour créer le tableau associatif.

Si vous créez un tableau associatif via un constructeur Object, vous profiter de l'initialisation de votre tableau à l'aide d'un littéral d'objet. Une occurrence de la classe Object, également appelée objet générique, présente le même fonctionnement qu'un tableau associatif. En fait, les occurrences Object sont essentiellement des tableaux associatifs. Vous pouvez utiliser ces tableaux pour des fonctions de type dictionnaire, lorsque les clés de chaîne sont plus pratiques que les index numériques. Chaque nom de propriété de l'objet générique devient la clé qui permet d'accéder à une valeur stockée. Pour plus d'informations sur les littéraux, consultez la section « Présentation des littéraux », à la page 99. Pour plus d'informations sur les classes, consultez le Chapitre 6, « Classes », à la page 197.

Pour créer un tableau associatif à l'aide d'un constructeur Object :

1. Créez un document Flash, puis enregistrez-le sous le nom **assocArray.fla**.
2. Ajoutez le code ActionScript suivant à l'Image 1 du scénario :

```
var monitorInfo:Object = {type:"Flat Panel", resolution:"1600 x 1200"};
trace(monitorInfo["type"] + ", " + monitorInfo["resolution"]);
```

   Ce code crée un tableau associatif appelé `monitorInfo` et utilise un littéral d'objet pour initialiser le tableau avec deux paires clé/valeur.

REMARQUE	Si vous n'avez pas besoin d'initialiser le tableau lors de la déclaration, vous pouvez utiliser le constructeur Object pour créer le tableau.

```
var monitorInfo:Object = new Object();
```

3. Choisissez Contrôle> Tester l'animation.

   Le panneau de sortie affiche le texte suivant :

   ```
   Flat Panel, 1600 x 1200
   ```

4. Ajoutez le code ActionScript suivant dans l'image 1 du scénario, à la suite du code saisi précédemment :

   ```
   monitorInfo["aspectRatio"] = "16:10";
   monitorInfo.colors = "16.7 million";
   trace(monitorInfo["aspectRatio"] + ", " + monitorInfo.colors);
   ```

   Après avoir créé le tableau à l'aide d'un littéral d'objet ou d'un constructeur de classe Object, vous pouvez lui ajouter de nouvelles valeurs en utilisant l'opérateur crochets (`[]`) ou l'opérateur point (`.`), comme l'illustre le code suivant. Le code que vous venez de saisir ajoute deux nouvelles valeurs au tableau `monitorInfo`.

5. Choisissez Contrôle > Tester l'animation.

   Le panneau de sortie affiche le texte suivant :

   ```
   16:10, 16.7 million
   ```

   Remarquez que les clés peuvent contenir des espaces. Cela est possible dans le cas de l'opérateur crochets, mais génère une erreur avec l'opérateur point. L'utilisation d'espace dans le nom de vos clés n'est donc pas conseillée. Pour plus d'informations sur les opérateurs crochets et point, consultez la section « Présentation des opérateurs », à la page 145. Pour plus d'informations sur le code correctement formaté, consultez la section « Mise en forme de la syntaxe ActionScript », à la page 750.

Le second moyen pour créer un tableau associatif consiste à utiliser le constructeur Array, puis l'opérateur crochets (`[]`) ou point (`.`) pour ajouter les paires de clé/valeur dans le tableau. Si vous déclarez votre tableau associatif comme étant de type Array, vous ne pouvez pas utiliser de littéral d'objet pour l'initialiser.

REMARQUE	L'utilisation du constructeur Array pour créer un tableau associatif ne présente aucun avantage. Ce constructeur convient mieux à la création des tableaux indexés.

L'exemple suivant présente l'utilisation du constructeur Array pour créer un tableau associatif.

Pour créer un tableau associatif à l'aide du constructeur Array :

1. Créez un document Flash, puis enregistrez-le sous le nom **assocArray2.fla**.
2. Ajoutez le code ActionScript suivant à l'Image 1 du scénario :

```
var monitorInfo:Array = new Array();
monitorInfo["type"] = "Flat Panel";
monitorInfo["resolution"] = "1600 x 1200";
trace(monitorInfo["type"] + ", " + monitorInfo["resolution"]);
```

   Ce code crée un tableau associatif appelé `monitorInfo` à l'aide du constructeur Array et ajoute les clés appelées `type` et `resolution`, avec leurs valeurs.

3. Choisissez Contrôle > Tester l'animation.

   Le panneau de sortie affiche le texte suivant :

```
Flat Panel, 1600 x 1200
```

REMARQUE	L'utilisation du constructeur Array pour créer un tableau associatif ne présente aucun avantage. Ce constructeur convient mieux à la création des tableaux indexés.

Les tableaux associatifs sont essentiellement des occurrences de la classe Object et l'utilisation du constructeur Array pour créer des tableaux associatifs ne présente aucun avantage. Même si vous créez un tableau associatif à l'aide du constructeur `new Array()`, vous ne pouvez utiliser aucune méthode ou propriété de la classe Array (telle que `sort()` ou `length`) lorsque vous utilisez un tableau associatif. Si vous souhaitez utiliser des paires clé/valeur au lieu d'un index numérique, vous devez utiliser la classe Object et non un tableau associatif.

Présentation des opérateurs

Cette section décrit des règles générales au sujet de types courants d'opérateurs, de la priorité des opérateurs et de leur associativité.

Les opérateurs sont des caractères qui spécifient comment combiner, comparer ou modifier les valeurs d'une expression. Une expression est une instruction que Flash peut évaluer et qui renvoie une valeur. Pour créer une expression, vous pouvez associer des opérateurs et des valeurs ou appeler une fonction. Pour plus d'informations sur les expressions, consultez la section « Présentation de la syntaxe, des instructions et des expressions », à la page 82.

Une expression mathématique, par exemple, utilise des opérateurs numériques pour manipuler les valeurs que vous utilisez. +, <, * et = sont des exemples d'opérateurs. Une expression se compose d'opérateurs et d'*opérandes*. Elle désigne toute combinaison valide de symboles ActionScript représentant une valeur. Un opérande est une partie du code sur laquelle l'opérateur exécute une action. Par exemple, dans l'expression `x + 2`, `x` et 2 sont des opérandes et + est un opérateur.

Dans votre code, vous utilisez fréquemment des expressions et des opérateurs. Pour créer une expression, vous pouvez associer des opérateurs et des valeurs ou appeler une fonction.

REMARQUE

Cette section décrit brièvement l'utilisation de chaque type d'opérateur. Pour plus d'informations sur chaque opérateur, y compris les opérateurs spéciaux qui ne font pas partie des catégories suivantes, consultez le *Guide de référence du langage ActionScript 2.0*.

Les parties de votre code sur lesquelles les opérateurs agissent sont appelés *opérandes*. Vous pouvez utiliser l'opérateur d'addition (+) pour ajouter des valeurs à un littéral numérique, par exemple pour ajouter la valeur d'une variable appelée `myNum`.

```
myNum + 3;
```

Dans cet exemple, `myNum` et `3` sont des opérandes.

Cette section décrit les règles générales qui régissent les types courants d'opérateurs, leur priorité et leur associativité.

Pour plus d'informations sur les opérateurs qui n'entrent pas dans ces catégories, consultez le *Guide de référence du langage ActionScript 2.0*, qui contient des informations sur tous les opérateurs disponibles.

Les sections suivantes présentent quelques utilisations courantes des opérateurs. Pour plus d'informations sur l'utilisation de plusieurs opérateurs dans un même code, consultez la section « Utilisation des opérateurs dans un document », à la page 168.

Utilisation d'opérateurs pour manipuler les valeurs

Les opérateurs sont souvent utilisés pour manipuler des valeurs dans Flash. Par exemple, vous pouvez créer un jeu dans lequel le score change selon les interactions de l'utilisateur avec les occurrences de la scène. Vous pouvez utiliser une variable pour stocker la valeur et des opérateurs pour manipuler la valeur de la variable.

Par exemple, vous pouvez augmenter la valeur d'une variable appelée `myScore`. L'exemple suivant décrit l'utilisation des opérateurs + (addition) et += (affectation d'addition) pour ajouter et incrémenter des valeurs de votre code.

Pour manipuler les valeurs à l'aide d'opérateurs :

1. Créez un document Flash.
2. Ouvrez le panneau Actions (Fenêtre > Actions) et saisissez le code suivant dans la fenêtre de script :

```
// Exemple un
var myScore:Number = 0;
myScore = myScore + 1;
trace("Example one: " + myScore); // 1

// Exemple deux
var secondScore:Number = 1;
secondScore += 3;
trace("Example two: " + secondScore); // 4
```

3. Choisissez Contrôle > Tester l'animation.

   Le panneau de sortie affiche le texte suivant :

   ```
   Example one: 1
   Example two: 4
   ```

   L'opérateur d'addition est très simple car il se contente d'additionner deux valeurs. Dans le premier exemple de code, il additionne la valeur actuelle de `myScore` avec le nombre 1, puis stocke le résultat dans la variable `myScore`.

Le deuxième exemple de code utilise l'opérateur d'affectation d'addition pour ajouter et affecter une nouvelle valeur dans la même étape. Vous pouvez réécrire la ligne `myScore = myScore + 1` (de l'exercice précédent) sous la forme `myScore++` ou encore `myScore += 1`. L'opérateur d'incrémentation (++) est un moyen simple de dire `myScore = myScore + 1`, car il traite simultanément une incrémentation et une affectation. Vous pouvez voir un exemple de ce type d'opérateur dans le code ActionScript suivant :

```
var myNum:Number = 0;
myNum++;
trace(myNum); // 1
myNum++;
trace(myNum); // 2
```

Notez que le code précédent ne contient pas d'opérateur d'affectation, mais repose sur un opérateur d'incrémentation.

Vous pouvez manipuler la valeur d'une variable à l'aide d'opérateurs tant qu'une condition est `true` (vraie). Par exemple, vous pouvez utiliser l'opérateur d'incrémentation (++) pour incrémenter la variable `i` tant que la condition est vraie. Dans le code suivant, la condition est `true` tant que `i` est inférieur à 10. Tant que cette condition est vraie, `i` est incrémenté d'une unité via `i++`.

```
var i:Number;
for (i = 1; i < 10; i++) {
  trace(i);
}
```

Le panneau de sortie affiche les nombres 1 à 9, c'est-à-dire `i` dont la valeur est incrémentée jusqu'à ce que la condition de fin soit atteinte (`i` égale 10). La dernière valeur affichée est 9. De ce fait, la valeur de `i` est 1 lorsque la lecture du fichier SWF commence, puis 9 à la fin du suivi.

Pour plus d'informations sur les conditions et les boucles, consultez la section « Présentation des instructions », à la page 109.

Priorité et associativité des opérateurs

Lorsque plusieurs opérateurs sont utilisés dans la même instruction, certains sont prioritaires par rapport à d'autres. La priorité et l'associativité des opérateurs déterminent leur ordre de traitement. La hiérarchie d'ActionScript détermine l'ordre d'exécution des opérateurs. Le tableau décrivant cette hiérarchie est présenté à la fin de cette section.

Bine qu'il soit évident, pour ceux qui connaissent bien l'arithmétique ou la programmation de base, que le compilateur traite l'opérateur de multiplication (*) avant l'opérateur d'addition (+), le compilateur a besoin d'instructions claires quant à l'ordre à appliquer aux opérateurs. L'ensemble de ces instructions est appelé *ordre de priorité des opérateurs*.

L'utilisation des opérateurs de multiplication et d'addition constitue un exemple de priorité des opérateurs :

```
var mySum:Number;
mySum = 2 + 4 * 3;
trace(mySum); // 14
```

Le résultat de cette instruction est 14, car la multiplication est prioritaire. Ainsi, 4 * 3 est d'abord calculé, puis 2 est ajouté au résultat.

Vous pouvez contrôler l'ordre des opérations en plaçant les expressions entre parenthèses. ActionScript définit une priorité par défaut que l'opérateur parenthèses (`()`) permet de modifier. Si l'addition est mise entre parenthèses, ActionScript l'effectue en premier :

```
var mySum:Number;
mySum = (2 + 4) * 3;
trace(mySum); // 18
```

Dans ce cas, le résultat est `18`.

Il est également possible que des opérateurs aient la même priorité. Dans ce cas, l'associativité détermine leur ordre d'exécution. L'associativité peut aller de gauche à droite ou de droite à gauche.

Examinons de nouveau l'opérateur de multiplication. Comme il présente une associativité de gauche à droite, les deux instructions sont identiques.

```
var mySum:Number;
var myOtherSum:Number;
mySum = 2 * 4 * 3;
myOtherSum = (2 * 4) * 3;
trace(mySum); // 24
trace(myOtherSum); // 24
```

Il arrive que plusieurs opérateurs de même priorité apparaissent dans la même expression. Dans ce cas, le compilateur utilise les règles d'*associativité* pour identifier le premier opérateur à traiter. Tous les opérateurs binaires, sauf les opérateurs d'affectation, sont *associatifs gauche*, ce qui signifie que les opérateurs de gauche sont traités avant ceux de droite. Les opérateurs d'affectation et l'opérateur conditionnel (`?:`) sont *associatifs droit*, ce qui signifie que les opérateurs de droite sont traités avant ceux de gauche. Pour plus d'informations sur l'affectation des opérateurs, consultez la section « Utilisation des opérateurs d'affectation », à la page 163. Pour plus d'informations sur l'opérateur conditionnel, consultez la section « Présentation de l'opérateur conditionnel », à la page 168.

Prenons par exemple les opérateurs inférieur à (<) et supérieur à (>), qui ont le même ordre de priorité. Lorsque les deux opérateurs sont employés dans la même expression, celui de gauche est traité en premier puisque tous deux sont associatifs gauche. Cela signifie que les deux instructions suivantes donnent le même résultat :

```
trace(3 > 2 < 1);   // false
trace((3 > 2) < 1); // false
```

L'opérateur supérieur à (>) est traité en premier, donnant un résultat `true` puisque l'opérande `3` est supérieur à l'opérande `2`. La valeur `true` est alors transmise à l'opérateur inférieur à (<), avec l'opérande `1`. L'opérateur inférieur à (<) convertit la valeur `true` en valeur numérique `1` et compare cette valeur au second opérande `1` pour renvoyer la valeur `false` (la valeur `1` n'est pas inférieure à `1`).

N'oubliez pas l'ordre des opérandes dans votre code ActionScript, en particulier lorsque vous élaborez des conditions complexes et que vous savez combien de fois elles sont vraies. Par exemple, si vous savez que `i` sera supérieur à 50 dans votre condition, vous devez commencer par écrire `i<50`. Ce chiffre sera ainsi vérifié en premier et la seconde condition écrite n'a pas besoin d'être vérifiée systématiquement.

Le tableau suivant présente les opérateurs d'ActionScript 2.0 par ordre décroissant de priorité. Chaque ligne du tableau contient des opérateurs de même priorité. Chaque ligne d'opérateurs a une priorité supérieure à la ligne située juste après dans le tableau. Pour plus d'informations et des recommandations d'utilisation des opérateurs et des parenthèses, consultez le Chapitre 17, « Mise en forme de la syntaxe ActionScript », à la page 750.

Groupe	opérateurs
Primaire	`[] {x:y} () f(x) new x.y x[y]`
Suffixe	`x++ x--`
Unaire	`++x --x + - ~ ! delete typeof void`
De multiplication	`* / %`
Additive	`+ -`

Groupe	opérateurs
Décalage au niveau du bit	`<< >> >>>`
Relationnels	`< > <= >= instanceof`
Egalité	`== != === !==`
AND au niveau du bit	`&`
XOR au niveau du bit	`^`
OR au niveau du bit	`|`
AND logique	`&&`
OR logique	`||`
Conditionnelle	`?:`
Affectation	`= *= /= %= += -= <<= >>= >>>= &= ^= |=`
Virgule	`,`

Utilisation d'opérateurs avec des chaînes

Si deux types de données sont différents, les opérateurs de comparaison convertissent le type d'un opérande pour le faire correspondre au type de l'autre opérande. Si un seul opérande est une chaîne et que l'autre est un nombre, ActionScript convertit l'opérande de chaîne en nombre et effectue une comparaison numérique. Cette règle a une exception : l'opérateur d'égalité stricte (===) qui se comporte de la même façon que l'opérateur d'égalité (==), à la différence que les types de données ne sont pas convertis. Le résultat est `true` lorsque les deux expressions sont égales, types de données inclus. Pour plus d'informations sur les opérateurs numériques, consultez la section « Utilisation des opérateurs numériques », à la page 157.

A l'exception de l'opérateur d'égalité (==), les opérateurs de comparaison (>, >=, < et <=) n'affectent pas les chaînes de la même façon que les autres valeurs.

Ces opérateurs comparent les chaînes pour déterminer celle qui apparaît en premier dans l'ordre alphabétique. Les chaînes en majuscules ont priorité sur celles qui sont en minuscules. Cela signifie que le terme « Egg » vient avant « chicken ».

```
var c:String = "chicken";
var e:String = "Egg";
trace(c < e); // false
var riddleArr:Array = new Array(c, e);
trace(riddleArr); // chicken,Egg
trace(riddleArr.sort()); // Egg,chicken
```

Dans ce code ActionScript, la méthode `sort()` de la classe Array trie le contenu du tableau dans l'ordre alphabétique. Vous voyez alors que la valeur « Egg » vient avant la valeur « chicken » car la majuscule E vient avant la minuscule c. Pour comparer des chaînes indépendamment de la casse, vous devez les convertir en majuscules ou en minuscules avant la comparaison. Pour plus d'informations sur les opérateurs de comparaison, consultez les sections « Présentation des opérateurs d'égalité », à la page 159 et « Utilisation des opérateurs relationnels et d'égalité », à la page 159.

Les méthodes `toLowerCase()` ou `toUpperCase()` permettent de convertir les chaînes en une même casse avant de les comparer. Dans l'exemple suivant, les deux chaînes sont converties en minuscules puis comparées. Le terme « chicken » vient maintenant avant « egg » :

```
var c:String = "chicken";
var e:String = "Egg";
trace(c.toLowerCase() < e.toLowerCase()); // true
```

REMARQUE

Les opérateurs de comparaison ne comparent que deux chaînes. Par exemple, ils ne comparent pas les valeurs lorsqu'un opérande est une valeur numérique. Si l'un des opérandes est une chaîne, ActionScript convertit les deux opérandes en nombres et effectue une comparaison numérique.

Les opérateurs permettent de manipuler des chaînes. Vous pouvez utiliser l'opérateur d'addition (+) pour concaténer deux opérandes de chaînes. Vous avez peut-être déjà fait cette opération si vous avez écrit des instructions `trace`. Par exemple, vous pouvez écrire le code suivant :

```
var myNum:Number = 10;
trace("The variable is " +  myNum + ".");
```

Lorsque vous testez le code, le résultat suivant s'affiche dans le panneau de sortie :

```
The variable is 10.
```

Dans l'exemple précédent, l'instruction `trace` utilise l'opérateur + pour concaténer au lieu d'additionner. Lorsque vous manipulez des chaînes et des nombres, Flash préfère parfois concaténer plutôt qu'additionner numériquement.

Par exemple, vous pouvez concaténer deux chaînes à partir de variables différentes dans un seul champ de texte. Dans le code ActionScript suivant, la variable `myNum` fait une concaténation avec une chaîne, et cette dernière affiche le champ `myTxt` sur la scène.

```
this.createTextField("myTxt", 11, 0, 0, 100, 20);
myTxt.autoSize = "left";
var myNum:Number = 10;
myTxt.text = "One carrot. " + myNum + " large eggplants.";
myTxt.text += " Lots of vegetable broth.";
```

Ce code donne le résultat suivant dans un champ de texte avec le nom d'occurrence myTxt :

```
One carrot. 10 large eggplants. Lots of vegetable broth.
```

L'exemple précédent décrit l'utilisation des opérateurs d'addition (+) et d'affectation d'addition (+=) pour concaténer les chaînes. Remarquez comment la troisième ligne de code utilise l'opérateur d'addition pour concaténer la valeur de la variable `myNum` dans le champ de texte, et comment la quatrième ligne utilise l'opérateur d'affectation d'addition pour concaténer une chaîne sur la valeur existante du champ de texte.

Si un seul opérande de chaîne de texte est une chaîne, Flash convertit l'autre opérande en chaîne. De ce fait, la valeur de `myNum` est convertie en une chaîne dans l'exemple précédent.

REMARQUE

ActionScript traite les espaces au début ou à la fin d'une chaîne comme faisant partie de la chaîne.

Utilisation des opérateurs point et d'accès au tableau

Vous pouvez utiliser les opérateurs point (`.`) et d'accès au tableau (`[]`) pour accéder aux propriétés ActionScript intégrées ou personnalisées. Vous utilisez les opérateurs point pour cibler certains index dans un objet. Par exemple, si l'un de vos objets contient des informations d'utilisateur, vous pouvez indiquer un certain nom de clé dans l'opérateur d'accès au tableau afin de récupérer le nom de l'utilisateur, comme l'illustre le code ActionScript suivant :

```
var someUser:Object = {name:"Hal", id:2001};
trace("User's name is: " + someUser["name"]); // Le nom de l'utilisateur
                                              // est : Hal
trace("User's id is: " + someUser["id"]); // L'ID de l'utilisateur est :
                                          // 2001
```

Par exemple, le code ActionScript suivant utilise l'opérateur point pour définir certaines propriétés à l'intérieur des objets :

```
myTextField.border = true;
year.month.day = 9;
myTextField.text = "My text";
```

Les opérateurs point et d'accès au tableau sont très similaires. L'opérateur point prend un identifiant en tant que propriété, mais l'opérateur d'accès au tableau évalue le contenu en nom, puis accède à la valeur de la propriété de ce nom. L'opérateur d'accès au tableau permet de définir et extraire de façon dynamique les variables et les noms d'occurrence.

L'opérateur d'accès au tableau se révèle très utile lorsque vous ignorez quelles clés sont stockées dans un objet. Dans ce cas, vous pouvez utiliser une boucle `for..in` pour effectuer une itération sur un objet ou un clip et afficher son contenu.

Pour utiliser les opérateurs point et d'accès au tableau :

1. Dans un nouveau document Flash, créez un clip sur le scénario principal.
2. Sélectionnez le clip et ouvrez l'inspecteur des propriétés.
3. Saisissez le nom d'occurrence de myClip.
4. Ajoutez le code ActionScript suivant à l'Image 1 du scénario :

```
myClip.spam = 5;
trace(myClip.spam); // 5
```

   Pour définir une valeur dans l'occurrence myClip du scénario actif, vous pouvez utiliser les opérateurs point ou d'accès au tableau, comme dans ce code ActionScript. Si vous écrivez une expression dans l'opérateur d'accès au tableau, ce dernier évalue d'abord l'expression, puis utilise le résultat comme nom de variable.

5. Sélectionnez Contrôle > Tester l'animation pour tester le document.

   Le panneau de sortie affiche 5.

6. Revenez dans l'environnement de programmation, puis remplacez la première ligne de code ActionScript par ce qui suit :

```
myClip["spam"] = 10;
```

7. Sélectionnez Contrôle > Tester l'animation pour tester le document.

   Le panneau de sortie affiche 10.

8. Revenez dans l'environnement de programmation et double-cliquez sur l'occurrence myClip.
9. Ajoutez quatre nouvelles occurrences dans l'occurrence myClip.
10. Utilisez l'inspecteur des propriétés pour donner les noms d'occurrences suivants aux quatre nouvelles occurrences : **nestedClip1**, **nestedClip2**, **nestedClip3**, **nestedClip4**.
11. Ajoutez le code suivant à l'image 1 du scénario principal :

```
var i:Number;
for (i = 1; i <= 4; i++) {
  myClip["nestedClip" + i]._visible = false;
}
```

   Ce code ActionScript désactive la visibilité de chaque clip imbriqué.

12. Choisissez Contrôle > Tester l'animation pour tester le code ActionScript que vous venez d'ajouter.

   Les quatre occurrences imbriquées sont à présent invisibles. Vous utilisez l'opérateur d'accès au tableau pour effectuer une itération sur chaque clip imbriqué dans l'occurrence myClip et définir sa propriété de visibilité dynamiquement. Vous gagnez ainsi du temps car vous n'avez plus besoin de cibler chaque occurrence de façon spécifique.

Vous pouvez également utiliser l'opérateur d'accès au tableau du côté gauche d'une affectation, ce qui vous permet de définir les noms d'occurrence, de variable et d'objet dynamiquement :

```
myNum[i] = 10;
```

Dans ActionScript 2.0, l'opérateur crochets permet d'accéder aux propriétés d'un objet créées dynamiquement, pour le cas où la définition de la classe de cet objet ne fournit pas l'attribut `dynamic`. Vous pouvez également créer des tableaux multidimensionnels à l'aide de cet opérateur. Pour plus d'informations sur la création de tableaux multidimensionnels à l'aide d'opérateurs d'accès au tableau, consultez la section « Création de tableaux multidimensionnels », à la page 138.

Présentation des opérateurs de suffixe

Les opérateurs de suffixe prennent un opérateur et incrémentent ou décrémentent sa valeur. Bien que ces opérateurs soient des opérateurs unaires, ils sont classés à part du fait de leur priorité supérieure et de leur comportement particulier. Pour plus d'informations sur les opérateurs unaires, consultez la section « Présentation des opérateurs unaires », à la page 156.

Lorsque vous utilisez un opérateur de suffixe dans une expression plus grande, la valeur de l'expression est renvoyée avant le traitement de cet opérateur. Par exemple, le code suivant montre comment la valeur de l'expression `xNum++` est renvoyée avant l'incrémentation de la valeur.

```
var xNum:Number = 0;
trace(xNum++); // 0
trace(xNum); // 1
```

Lorsque vous suivez ce code, le texte suivant s'affiche dans le panneau de sortie :

```
0
1
```

Les opérateurs de ce tableau ont le même ordre de priorité.

Opérateur	Opération effectuée
`++`	Incrémentation (suffixe)
`--`	Décrémentation (suffixe)

Présentation des opérateurs unaires

Les opérateurs unaires prennent un opérande. Les opérateurs d'incrémentation (++) et de décrémentation (--) de ce groupe sont des opérateurs de *préfixe*, c'est-à-dire qu'ils apparaissent avant l'opérande dans une expression. Ils peuvent aussi apparaître après l'opérande, auquel cas il s'agit d'opérateurs de *suffixe*. Pour plus d'informations sur les opérateurs de suffixe, consultez la section « Présentation des opérateurs de suffixe », à la page 155.

Les opérateurs de préfixe diffèrent de leur équivalent suffixe car l'opération d'incrémentation ou de décrémentation est effectuée avant le renvoi de la valeur de l'expression globale. Par exemple, le code suivant montre comment la valeur de l'expression `xNum++` est renvoyée avant l'incrémentation de la valeur.

```
var xNum:Number = 0;
trace(++xNum); // 1
trace(xNum); // 1
```

Tous les opérateurs de ce tableau ont le même ordre de priorité.

Opérateur	Opération effectuée
`++`	Incrémentation (préfixe)
`--`	Décrémentation (préfixe)
`+`	Unaire +
`!`	Unaire - (négation)
`typeof`	Renvoie les informations de type
`void`	Renvoie une valeur non définie

Présentation des opérateurs de multiplication

Les opérateurs de multiplication prennent deux opérandes et effectuent des multiplications, des divisions ou des calculs de modulo. Les autres opérateurs numériques comprennent les opérateurs d'ajout. Pour plus d'informations sur les opérateurs d'ajout, consultez la section « Présentation des opérateurs d'ajout », à la page 157.

Tous les opérateurs de ce tableau ont le même ordre de priorité.

Opérateur	Opération effectuée
`*`	Multiplication
`/`	Division
`%`	Modulo

Pour plus d'informations sur les opérateurs de multiplication, consultez la section « Utilisation des opérateurs numériques », à la page 157.

Présentation des opérateurs d'ajout

Les opérateurs d'ajout prennent deux opérandes et effectuent des calculs d'addition ou de soustraction. Les autres opérateurs numériques comprennent les opérateurs de multiplication. Pour plus d'informations sur les opérateurs de multiplication, consultez la section « Présentation des opérateurs de multiplication », à la page 156.

Les opérateurs de ce tableau ont le même ordre de priorité.

Opérateur	Opération effectuée
+	Addition
-	Soustraction

Pour plus d'informations sur l'utilisation des opérateurs d'ajout, consultez la section « Utilisation des opérateurs numériques », à la page 157.

Utilisation des opérateurs numériques

Les opérateurs numériques permettent d'ajouter, de soustraire, de diviser et de multiplier des valeurs dans ActionScript. Vous pouvez effectuer différentes sortes d'opérations arithmétiques. Le plus courant est l'opérateur d'incrémentation, généralement présenté sous la forme `i++`. Cet opérateur permet d'effectuer d'autres opérations. Pour plus d'informations sur l'opérateur d'incrémentation, consultez la section « Utilisation d'opérateurs pour manipuler les valeurs », à la page 147.

L'incrémentation peut être placé avant (*pré-incrémentation*) ou après (*post-incrémentation*) une opérande.

Pour comprendre les opérateurs numériques du langage ActionScript :

1. Créez un document Flash.
2. Saisissez le code ActionScript suivant sur l'image 1 du scénario :

```
// Exemple un
var firstScore:Number = 29;
if (++firstScore >= 30) {
  // devrait suivre
  trace("Success! ++firstScore is >= 30");
}
// Exemple deux
var secondScore:Number = 29;
if (secondScore++ >= 30) {
  // ne devrait pas suivre
  trace("Success! secondScore++ is >= 30");
}
```

3. Choisissez Contrôle > Tester l'animation pour tester le code ActionScript.

   Le bloc de code « Exemple un » suit, mais pas celui de « Exemple deux ». Le premier exemple utilise une pré-incrémentation (`++firstScore`) pour incrémenter et calculer `firstScore` avant sa vérification par rapport à 30. De ce fait, `firstScore` est incrémenté à 30, puis testé par rapport à 30.

   L'exemple deux utilise lui une post-incrémentation (`secondScore++`), c'est-à-dire que l'évaluation est effectuée après le test. Ainsi, 29 est comparé à 30, puis incrémenté à 30 après l'évaluation.

L'utilisation de l'opérateur d'addition peut donner lieu à des résultats inattendus si vous tentez d'ajouter des valeurs dans une expression, comme le montre l'exemple suivant :

```
trace("the sum of 5 + 2 is: " + 5 + 2); // La somme de 5 + 2 est : 52
```

Flash concatène les valeurs `5` et `2` au lieu de les additionner. Pour contourner le problème, vous pouvez placer l'expression `5+2` entre parenthèses, comme dans le code suivant :

```
trace("the sum of 5 + 2 is: " + (5 + 2)); // La somme de 5 + 2 est : 7
```

Pour plus d'informations sur la priorité des opérateurs, consultez la section « Priorité et associativité des opérateurs », à la page 149.

Lorsque vous transférez des données à partir de sources externes (tels que des fichiers XML, FlashVars, des services Web, etc.), faites très attention en utilisant les opérateurs numériques. Flash traite parfois les nombres comme des chaînes car le fichier SWF n'est pas conscient de leur type de données. Ainsi, l'addition de 3 et 7 pourrait donner 37 car les deux nombres peuvent être concaténés comme des chaînes au lieu d'être additionnés numériquement. Dans ce cas, vous devez convertir manuellement les données de chaînes en nombres via la fonction `Number()`.

Présentation des opérateurs relationnels

Les opérateurs relationnels prennent deux opérandes, comparent leur valeurs et renvoient une valeur booléenne. Tous les opérateurs de ce tableau ont le même ordre de priorité.

Opérateur	Opération effectuée
`<`	Inférieur à
`>`	Supérieur à
`<=`	Inférieur ou égal à
`>=`	Supérieur ou égal à
`instanceof`	Vérifie la chaîne prototype
`in`	Vérifie les propriétés des objets

Pour plus d'informations sur l'utilisation des opérateurs relationnels, consultez la section « Utilisation des opérateurs relationnels et d'égalité », à la page 159.

Présentation des opérateurs d'égalité

Les opérateurs d'égalité prennent deux opérandes, comparent leur valeurs et renvoient une valeur booléenne. Tous les opérateurs de ce tableau ont le même ordre de priorité.

Opérateur	Opération effectuée
==	Egalité
!=	Inégalité
===	Egalité stricte
!==	Inégalité stricte

Pour plus d'informations sur l'utilisation des opérateurs d'égalité, consultez la section « Utilisation des opérateurs relationnels et d'égalité », à la page 159.

Utilisation des opérateurs relationnels et d'égalité

Ces opérateurs, également appelés *opérateurs de comparaison*, comparent les valeurs des expressions et renvoient une valeur booléenne : `true` (vrai) ou `false` (faux). Ces opérateurs sont fréquemment employés dans des boucles et des instructions conditionnelles pour spécifier la condition de fin de la boucle.

Vous pouvez utiliser l'opérateur d'égalité (==) pour déterminer si les valeurs ou les références de deux opérandes sont égales et si cette comparaison renvoie une valeur booléenne. Les valeurs d'opérandes de type chaîne, nombre ou booléen effectuent la comparaison à l'aide d'une valeur. Les opérandes de type objet et tableau sont comparés par une référence.

Dans cet exemple, vous pouvez observer l'utilisation de l'opérateur d'égalité pour tester la longueur du tableau et afficher un message dans le panneau de sortie lorsque le tableau ne contient pas d'élément.

```
var myArr:Array = new Array();
if (myArr.length == 0) {
  trace("the array is empty.");
}
```

Lorsque vous choisissez Contrôle > Tester l'animation, la chaîne `the array is empty` (le tableau est vide) s'affiche dans le panneau de sortie.

Vous pouvez utiliser l'opérateur d'égalité pour comparer des valeurs, mais pas pour les définir. Une erreur courante consiste à utiliser l'opérateur d'affectation (=) pour contrôler l'égalité.

Pour utiliser les opérateurs relationnels et d'égalité dans votre code :

1. Créez un document Flash.
2. Saisissez le code ActionScript suivant sur l'image 1 du scénario :

```
var myNum:Number = 2;
if (myNum == 2) {
  // action
  trace("It equals 2");
}
```

   Dans ce code ActionScript, vous utilisez l'opérateur d'égalité (==) pour vérifier l'égalité. Vous vérifiez si la variable `myNum` est égale à 2.

3. Choisissez Contrôle > Tester l'animation.

   La chaîne `It equals 2` (est égal à 2) s'affiche dans le panneau de sortie.

4. Revenez dans l'environnement de programmation et modifiez :

```
var myNum:Number = 2;
```

   en :

```
var myNum:Number = 4;
```

5. Choisissez de nouveau Contrôle > Tester l'animation.

   La chaîne `It equals 2` (est égal à 2) ne s'affiche pas dans le panneau de sortie.

6. Revenez dans l'environnement de programmation et modifiez :

```
if (myNum == 2) {
```

   en

```
if (myNum = 2) {
```

7. Choisissez de nouveau Contrôle > Tester l'animation.

   La chaîne `It equals 2` (est égal à 2) s'affiche de nouveau dans le panneau de sortie.

   A l'étape 6, vous affectez la valeur `2` à `myNum`, au lieu de comparer `myNum` à `2`. Dans ce cas, l'instruction `if` s'exécute quelle que soit la valeur précédente de `myNum`, ce qui peut donner des résultats imprévus lors du test du document Flash.

   Pour plus d'informations sur l'utilisation appropriée de l'opérateur d'affectation, consultez la section « Utilisation des opérateurs d'affectation », à la page 163.

L'opérateur d'égalité stricte (===) est similaire à l'opérateur d'égalité, sauf qu'il n'effectue pas de conversion de type. Si les deux opérandes sont de types différents, l'opérateur d'égalité renvoie `false` (faux). L'opérateur d'inégalité stricte (!==) renvoie l'inverse de l'opérateur d'égalité stricte.

Le code ActionScript suivant présente la principale différence entre l'opérateur d'égalité (==) et l'opérateur d'égalité stricte (===) :

```
var num1:Number = 32;
var num2:String = new String("32");
trace(num1 == num2); // true
trace(num1 === num2); // false
```

D'abord, vous définissez les variables numériques : `num1` et `num2`. Si vous comparez les variables à l'aide de l'opérateur d'égalité, Flash tente de convertir les valeurs dans le même type de données, puis les compare afin de vérifier leur égalité. Lorsque vous utilisez l'opérateur d'égalité stricte (===), Flash n'effectue aucune conversion de type de données avant de comparer les valeurs. En résultat, Flash voit les variables comme deux valeurs distinctes.

Dans l'exemple suivant, vous allez utiliser l'opérateur supérieur ou égal à (>=) pour comparer les valeurs et exécuter le code en fonction de la valeur saisie par l'utilisateur dans un champ de texte.

Pour utiliser l'opérateur supérieur ou égal à dans votre code :

1. Sélectionnez Fichier > Nouveau, puis sélectionnez Document Flash pour créer un nouveau fichier FLA.
2. Ajoutez le code suivant à l'image 1 du scénario principal :

```
this.createTextField("myTxt", 20, 0, 0, 100, 20);
myTxt.type = "input";
myTxt.border = true;
myTxt.restrict = "0-9";

this.createEmptyMovieClip("submit_mc", 30);
submit_mc.beginFill(0xFF0000);
submit_mc.moveTo(0, 0);
submit_mc.lineTo(100, 0);
submit_mc.lineTo(100, 20);
submit_mc.lineTo(0, 20);
submit_mc.lineTo(0, 0);
submit_mc.endFill();
submit_mc._x = 110;

submit_mc.onRelease = function(evt_obj:Object):Void {
  var myNum:Number = Number(myTxt.text);
  if (isNaN(myNum)) {
    trace("Please enter a number");
    return;
  }
  if (myNum >= 10) {
    trace("Your number is greater than or equal to 10");
  } else {
    trace("Your number is less than 10");
  }
};
```

3. Choisissez Contrôle > Tester l'animation pour tester le code ActionScript.

   Vous pouvez également vérifier si certaines conditions sont vraies, puis exécuter un autre bloc si la condition ne l'est pas.

4. Modifiez la condition de votre code ActionScript de la façon suivante.

```
if (myNum == 10) {
  trace("Your number is 10");
} else {
    trace("Your number is not 10");
}
```

5. Choisissez Contrôle > Tester l'animation pour tester de nouveau le code ActionScript.

A l'exception de l'opérateur d'égalité stricte (===), les opérateurs de comparaison ne comparent des chaînes que si les deux opérandes sont des chaînes. Si un seul opérande est une chaîne, les deux opérandes sont convertis en nombres et une comparaison numérique est effectuée. Pour plus d'informations sur les chaînes et les opérateurs, consultez la section « Utilisation d'opérateurs avec des chaînes », à la page 151. Pour plus d'informations sur l'impact de l'ordre et de la priorité des opérateurs sur votre code ActionScript, consultez la section « Priorité et associativité des opérateurs », à la page 149.

Présentation des opérateurs d'affectation

Les opérateurs d'affectation prennent deux opérandes et affectent une valeur à l'un d'eux en fonction de la valeur de l'autre. Tous les opérateurs de ce tableau ont le même ordre de priorité.

Opérateur	Opération effectuée
=	Affectation
*=	Affectation de multiplication
/=	Affectation de division
%=	Affectation modulo
+=	Affectation d'addition
-=	Affectation de soustraction
<<=	Affectation de décalage gauche au niveau du bit
>>=	Affectation de décalage droit au niveau du bit
>>>=	Affectation de décalage droit au niveau du bit non signé
&=	Affectation AND au niveau du bit
^=	Affectation XOR au niveau du bit
|=	Affectation OR au niveau du bit

Pour plus d'informations sur l'utilisation des opérateurs d'affectation, consultez la section « Utilisation des opérateurs d'affectation », à la page 163.

Utilisation des opérateurs d'affectation

L'opérateur d'affectation (=) permet d'affecter une valeur donnée à une variable. Vous pouvez affecter une chaîne à une variable, comme suit :

```
var myText:String = "ScratchyCat";
```

Vous pouvez également utiliser l'opérateur d'affectation pour affecter plusieurs variables dans la même expression. Dans l'instruction suivante, la valeur 10 est affectée aux variables `numOne`, `numTwo` et `numThree`.

```
var numOne:Number;
var numTwo:Number;
var numThree:Number;
numOne = numTwo = numThree = 10;
```

Vous pouvez aussi utiliser des opérateurs d'affectation composés pour combiner des opérations. Ces opérateurs agissent sur les deux opérandes, puis affectent la nouvelle valeur au premier. Par exemple, les deux instructions suivantes donnent le même résultat :

```
var myNum:Number = 0;
myNum += 15;
myNum = myNum + 15;
```

Présentation des opérateurs logiques

Les opérateurs logiques comparent des valeurs booléennes (`true` et `false`) et renvoient une troisième valeur booléenne dépendant de la comparaison. Par exemple, si deux opérandes renvoient `true`, l'opérateur logique AND (`&&`) renvoie `true`. Si l'un des opérandes, ou les deux, renvoie(nt) `true`, l'opérateur logique OR (`||`) renvoie `true`.

Les opérateurs logiques prennent deux opérandes et renvoient une valeur booléenne. L'ordre de priorité de ces opérateurs diffère et ils sont présentés dans le tableau suivant par ordre décroissant de priorité :

Opérateur	Opération effectuée
`&&`	AND logique
`||`	OR logique

Pour plus d'informations sur l'utilisation des opérateurs logiques, consultez la section « Utilisation des opérateurs logiques », à la page 164.

Utilisation des opérateurs logiques

Les opérateurs logiques sont souvent utilisés en complément des opérateurs de comparaison pour déterminer la condition d'une instruction `if`, comme dans l'exemple suivant.

Pour utiliser des opérateurs logiques dans votre code :

1. Choisissez Fichier > Nouveau et créez un document Flash.
2. Dans le panneau Actions, entrez le code ActionScript suivant sur l'image 1 du scénario :

```
this.createTextField("myTxt", 20, 0, 0, 100, 20);
myTxt.type = "input";
myTxt.border = true;
myTxt.restrict = "0-9";

this.createEmptyMovieClip("submit_mc", 30);
submit_mc.beginFill(0xFF0000);
submit_mc.moveTo(0, 0);
submit_mc.lineTo(100, 0);
submit_mc.lineTo(100, 20);
submit_mc.lineTo(0, 20);
submit_mc.lineTo(0, 0);
submit_mc.endFill();
submit_mc._x = 110;

submit_mc.onRelease = function():Void {
  var myNum:Number = Number(myTxt.text);
  if (isNaN(myNum)) {
    trace("Please enter a number");
    return;
  }
  if ((myNum > 10) && (myNum < 20)) {
    trace("Your number is between 10 and 20");
  } else {
    trace("Your number is NOT between 10 and 20");
  }
};
```

Dans ce code ActionScript, créez un champ de texte au moment de l'exécution. Si vous entrez un nombre dans ce champ de texte, puis cliquez sur un bouton de la scène, Flash utilise l'opérateur logique pour afficher un message dans le panneau de sortie. Le message dépend de la valeur du nombre saisi dans le champ.

Lorsque vous utilisez des opérandes, prenez garde à l'ordre employé, en particulier avec des conditions complexes. Le code suivant présente l'utilisation de l'opérateur logique AND pour vérifier qu'un nombre est compris entre 10 et 20. Selon le résultat, un message approprié s'affiche. Si le nombre est inférieur à 10 ou supérieur à 20, un autre message s'affiche dans le panneau de sortie.

```
submit_mc.onRelease = function():Void {
  var myNum:Number = Number(myTxt.text);
  if (isNaN(myNum)) {
    trace("Please enter a number");
    return;
  }
  if ((myNum > 10) && (myNum < 20)) {
    trace("Your number is between 10 and 20");
  } else {
    trace("Your number is NOT between 10 and 20");
  }
};
```

Présentation des opérateurs de décalage au niveau du bit

Ces opérateurs prennent deux opérandes et décalent les bits du premier selon la valeur spécifiée dans le second. Tous les opérateurs de ce tableau ont le même ordre de priorité.

Opérateur	Opération effectuée
<<	Décalage gauche au niveau du bit
>>	Décalage droit au niveau du bit
>>>	Décalage droit non signé au niveau du bit

Pour plus d'informations sur l'utilisation des opérateurs au niveau du bit, consultez la section « Utilisation des opérateurs au niveau du bit », à la page 166. Pour des informations spécifiques sur chaque opérateur au niveau du bit, consultez la section correspondante du *Guide de référence du langage ActionScript 2.0*.

Présentation des opérateurs logiques au niveau du bit

Ces opérateurs prennent deux opérandes et effectuent des opérations logiques au niveau des bits. La priorité de ces opérateurs diffère et ils sont présentés dans le tableau suivant par ordre décroissant de priorité :

Opérateur	Opération effectuée
`&`	AND au niveau du bit
`^`	XOR au niveau du bit
`|`	OR au niveau du bit

Pour plus d'informations sur l'utilisation des opérateurs au niveau du bit, consultez la section « Utilisation des opérateurs au niveau du bit », à la page 166. Pour des informations spécifiques sur chaque opérateur au niveau du bit, consultez la section correspondante du *Guide de référence du langage ActionScript 2.0*.

Utilisation des opérateurs au niveau du bit

Les opérateurs au niveau du bit manipulent (en interne) les nombres à virgule flottante pour les transformer en entiers 32 bits. L'opération exacte dépend de l'opérateur, mais toutes les opérations au niveau du bit évaluent chaque bit d'un entier 32 bits séparément pour calculer une nouvelle valeur. Pour obtenir la liste des opérateurs de décalage au niveau du bit, consultez la section « Présentation des opérateurs de décalage au niveau du bit », à la page 165. Pour obtenir la liste des opérateurs logiques au niveau du bit, consultez la section « Présentation des opérateurs logiques au niveau du bit », à la page 166.

L'utilisation d'opérateurs au niveau du bit dans Flash n'est pas très fréquente, mais peut se révéler très utile dans certains cas. Par exemple, vous pouvez créer une matrice d'autorisations pour un projet Flash, sans créer de variables distinctes pour chaque type d'autorisation. Dans ce cas, vous pouvez utiliser des opérateurs du niveau du bit.

L'exemple suivant décrit l'utilisation de l'opérateur OR au niveau du bit avec la méthode `Array.sort()` pour spécifier les options de tri.

Pour utiliser l'opérateur OR au niveau du bit :

1. Choisissez Fichier > Nouveau et créez un document Flash.
2. Saisissez le code ActionScript suivant dans le panneau Actions :

```
var myArr:Array = new Array("Bob", "Dan", "doug", "bill", "Hank",
  "tom");
trace(myArr); // Bob,Dan,doug,bill,Hank,tom
myArr.sort(Array.CASEINSENSITIVE | Array.DESCENDING);
trace(myArr); // tom,Hank,doug,Dan,Bob,bill
```

   La première ligne définit un tableau de noms aléatoires et les suit dans le panneau de sortie. Vous appelez ensuite la méthode `Array.sort()` et spécifiez deux options de tri à l'aide des valeurs constantes `Array.CASEINSENSITIVE` et `Array.DESCENDING`. Cette méthode trie les éléments du tableau en ordre inverse (de z à a). La recherche ne respecte pas la casse. Les lettres `a` et `A` sont traitées à l'identique, plutôt que d'obtenir un `Z` avant un `a`.

3. Choisissez Contrôle > Tester l'animation pour tester votre code ActionScript. Le texte suivant apparaît dans le panneau de sortie :

```
Bob,Dan,doug,bill,Hank,tom
tom,Hank,doug,Dan,Bob,bill
```

Cinq options sont disponibles dans la méthode de tri :

- 1 ou Array.CASEINSENSITIVE (binaire = 1)
- 2 ou Array.DESCENDING (binaire = 10)
- 4 ou Array.UNIQUESORT (binaire = 100)
- 8 ou Array.RETURNINDEXEDARRAY (binaire = 1000)
- 16 ou Array.NUMERIC (binaire = 10000)

Trois méthodes différentes permettent de définir les options de tri d'un tableau :

```
my_array.sort(Array.CASEINSENSITIVE | Array.DESCENDING); // Constantes
my_array.sort(1 | 2); // Nombres
my_array.sort(3); // Ajout de nombres
```

Bien que cela ne soit pas immédiatement évident, les valeurs des nombres des options de tri sont en fait des chiffres au niveau du bit (binaire ou base 2). La valeur de la constante `Array.CASEINSENSITIVE` est égale à la valeur numérique 1, correspondant également à la valeur binaire de 1. La valeur de la constante `Array.DECENDING` est égale à la valeur numérique 2 ou la valeur binaire 10.

L'utilisation des nombres binaires prête parfois à confusion. Ils n'ont que deux valeurs possibles, 1 ou 0, raison pour laquelle 2 est représenté par 10. Sous forme binaire, le nombre 3 devient 11 (1+10), le nombre 4 devient 100, 5 devient 101, etc.

Le code ActionScript suivant présente le tri d'un tableau de valeurs numériques par ordre décroissant via l'utilisation de l'opérateur AND au niveau du bit pour additionner les constantes `Array.DESCENDING` et `Array.NUMERIC`.

```
var scores:Array = new Array(100,40,20,202,1,198);
trace(scores); // 100,40,20,202,1,198
trace(scores.sort()); // 1,100,198,20,202,40
var flags:Number = Array.NUMERIC|Array.DESCENDING;
trace(flags); // 18 (base 10)
trace(flags.toString(2)); // 10010 (binaire -- base 2)
trace(scores.sort(flags)); // 202,198,100,40,20,1
```

Présentation de l'opérateur conditionnel

L'opérateur conditionnel est un opérateur ternaire, c'est-à-dire qu'il prend trois opérandes. Il correspond à une méthode abrégée de l'application de l'instruction conditionnelle `if..else` :

Opérateur	Opération effectuée
`?:`	Conditionnelle

Pour plus d'informations sur l'utilisation des opérateurs conditionnels, consultez la section « Opérateur conditionnel et syntaxe de remplacement », à la page 121.

Utilisation des opérateurs dans un document

Dans l'exemple suivant, vous utilisez la méthode `Math.round()` pour arrondir des calculs selon un nombre arbitraire de décimales. Cette méthode arrondit la valeur du paramètre `score` à l'entier immédiatement supérieur ou inférieur et renvoie la valeur. En modifiant légèrement le code ActionScript, vous pouvez faire en sorte que Flash arrondisse les résultats à un certain nombre de décimales.

Dans l'exemple, vous utilisez également les opérateurs de division et de multiplication pour calculer la note de l'utilisateur en divisant le nombre de réponses correctes par le nombre total de questions. La note de l'utilisateur peut ensuite être multipliée par un certain nombre et être présentée sous forme de pourcentage entre 0 % et 100 %. Vous utilisez ensuite l'opérateur d'addition pour concaténer la note de l'utilisateur dans une chaîne affichée dans le panneau de sortie.

Pour utiliser des opérateurs dans votre code ActionScript :

1. Créez un document Flash.
2. Saisissez le code ActionScript suivant sur l'image 1 du scénario principal :

```
var correctAnswers:Number = 11;
var totalQuestions:Number = 13;
// Arrondi au nombre entier le plus proche
// var score:Number = Math.round(correctAnswers / totalQuestions * 100);
// Arrondi à deux chiffres après la virgule
var score:Number = Math.round(correctAnswers / totalQuestions * 100 *
  100) / 100;
trace("You got " + correctAnswers + " out of " + totalQuestions + "
  answers correct, for a score of " + score + "%.");
```

3. Choisissez Contrôle> Tester l'animation.

   Le panneau de sortie affiche le texte suivant :

   ```
   You got 11 out of 13 answers correct, for a score of 84.62%.
   ```

   Lorsque vous appelez la méthode `Math.round()` dans cet exemple, la note est arrondie au nombre entier le plus proche (85) et s'affiche dans le panneau de sortie. Si vous multipliez le nombre par 100 avant d'appeler la méthode `Math.round()`, puis le divisez par 100, Flash arrondit la note à deux chiffres après la virgule. Ainsi, la note est plus précise.
4. Affectez la valeur 3 à la variable `correctAnswers`, puis choisissez Contrôle > Tester l'animation pour tester de nouveau le fichier SWF.

Si vous développez une application de test, vous pouvez créer une série de questions vrai/faux ou à choix multiples à l'aide des composants RadioButton et Label. Lorsque l'utilisateur a terminé et clique sur le bouton pour soumettre son questionnaire rempli, vous pouvez comparer ses réponses aux clés de correction, puis calculer sa note.

CHAPITRE 5

Fonctions et méthodes

5

Lorsque vous créez du code et des classes ActionScript et utilisez des méthodes, il est important de bien comprendre les fonctions. Vous allez exploiter plusieurs sortes de fonctions. Ce chapitre vous permet de découvrir les fonctions et les méthodes : comment les utiliser dans vos applications lorsque vous avez recours à des classes intégrées et comment les écrire. Dans le Chapitre 6, « Classes », vous allez créer des classes personnalisées dans lesquelles vous écrirez régulièrement des fonctions. Vous découvrirez également comment écrire des fonctions dans des fichiers de classe ActionScript.

Vous pouvez utiliser des fonctions dans votre code pour ajouter de l'interactivité, des animations et d'autres effets à vos applications. Ce chapitre traite des types de fonction que vous pouvez écrire dans vos applications Flash. Pour obtenir des d'informations sur les fonctions et les méthodes, ainsi que des exercices dans lesquels vous pouvez écrire et utiliser des fonctions et des méthodes dans Flash, consultez les rubriques suivantes :

Présentation des fonctions et des méthodes

Ces éléments sont des blocs de code ActionScript que vous pouvez réutiliser n'importe où dans un fichier SWF. Vous pouvez écrire une fonction dans le fichier FLA ou dans un fichier ActionScript externe, puis l'appeler depuis tout emplacement de vos documents.
Les méthodes sont simplement des fonctions placées dans la définition d'une classe ActionScript. Vous pouvez définir des fonctions pour exécuter une série d'instructions sur des valeurs transmises. Vos fonctions peuvent également renvoyer des valeurs. Une fois définie, une fonction peut être appelée à partir de tout scénario, y compris celui d'un fichier SWF chargé.

Si vous transmettez des valeurs en tant que paramètres à une fonction, cette dernière peut exécuter des calculs à l'aide de ces valeurs. Chaque fonction possède ses propres caractéristiques. Certaines vous demandent de transmettre des types de valeurs ou certaines valeurs. Si vous transmettez à une fonction plus de valeurs qu'elle n'en réclame, les valeurs excédentaires sont ignorées. Si vous omettez un paramètre obligatoire, la fonction lui attribue le type de données `non défini`. Cet oubli peut provoquer des erreurs au moment de l'exécution. Une fonction peut également renvoyer des valeurs (voir « Renvoi de valeurs depuis les fonctions », à la page 192).

REMARQUE	Pour appeler une fonction, sa définition doit se trouver dans l'image que la tête de lecture a atteinte.

Une fonction bien rédigée peut être considérée comme une « boîte noire ». Si cette fonction contient des commentaires pertinents et judicieusement placés, au sujet de son entrée, sa sortie et son rôle, toute personne qui l'utilise n'a pas nécessairement besoin de comprendre son fonctionnement interne.

La syntaxe de base d'une simple *fonction nommée* est la suivante :

```
function traceMe() {
  trace("your message");
}
traceMe();
```

Pour plus d'informations sur l'écriture des fonctions nommées, consultez la section « Ecriture des fonctions nommées », à la page 177.

La syntaxe de base d'une simple fonction nommée se basant sur l'exemple précédent en transmettant un paramètre, `yourMessage`, est la suivante :

```
function traceMe(yourMessage:String) {
  trace(yourMessage);
}
traceMe("How you doing?");
```

Si vous souhaitez transmettre plusieurs paramètres, vous pouvez utiliser le code suivant :

```
var yourName:String = "Ester";
var yourAge:String = "65";
var favSoftware:String = "Flash";
function traceMe(favSoftware:String, yourName:String, yourAge:String) {
  trace("I'm " + yourName + ", I like " + favSoftware + ", and I'm " +
  yourAge + ".");
}
traceMe(favSoftware,yourName,yourAge);
```

Pour plus d'informations sur la transmission de paramètres, consultez la section « Transmission de paramètres à une fonction », à la page 190.

Vous pouvez écrire de nombreux types de fonction. Pour plus d'informations sur l'écriture de fonctions et pour obtenir des liens vers des sections sur l'écriture de types de fonction particuliers, consultez la section « Types de méthodes et de fonctions », à la page 173. Pour un exemple comparatif entre les méthodes et les fonctions, consultez la section « Fonctionnement des méthodes », à la page 194.

REMARQUE

Pour plus d'informations sur l'écriture de code à l'aide de l'Assistant de script, consultez le guide *Utilisation de Flash*.

Pour plus d'informations sur les fonctions et les méthodes, consultez les rubriques suivantes :

- « Types de méthodes et de fonctions », à la page 173

Types de méthodes et de fonctions

Les fonctions qui appartiennent à une classe sont considérées comme les *méthodes* de cette classe. Dans vos applications, vous pouvez utiliser plusieurs types de fonction, dont des fonctions intégrées, des fonctions nommées et définies par l'utilisateur, des fonctions anonymes, des fonctions de rappel, des fonctions constructeur et des littéraux de fonction. Les sections suivantes contiennent des informations sur la façon de définir ces fonctions.

Vous pouvez également écrire des fonctions dans un fichier ActionScript. Vous les utilisez comme méthodes dans vos scripts. Dans l'exemple suivant, la classe Person affiche une méthode de constructeur et des méthodes de classe, d'occurrence et d'accesseur (lecture et définition). Les commentaires contenus dans cet exemple de code indiquent où ont lieu ces méthodes dans le code.

REMARQUE

Pour plus d'informations sur la rédaction de fichiers de classe comme celui-ci, consultez le Chapitre 6, « Classes », à la page 197.

```
class Personne {
  public static var numPeople:Number = 0;

  // Membres d'occurrence
  private var _speed:Number;

  // Constructeur
  public function Person(speed:Number) {
    Person.numPeople++;
    this._speed = speed;
  }

  // Méthodes statiques
  public static function getPeople():Number {
    return Person.numPeople;
  }

  // Méthodes d'occurrence
  public function walk(speed:Number):Void {
    this._speed = speed;
  }
  public function run():Void {
    this._speed *= 2;
  }
  public function rest():Void {
    this._speed = 0;
  }

  // Lecture/définition (méthodes d'accesseur)
  public function get speed():Number {
    return this._speed;
  }
}
```

Pour consulter une démonstration complète sur la façon de rédiger des méthodes comme celle de l'exemple de code précédent, consultez le Chapitre 6, « Classes », à la page 197.
Les méthodes utilisées dans votre code peuvent appartenir à une classe intégrée au langage ActionScript. MovieClip et Math sont des exemples de classes de niveau supérieur que vous pouvez utiliser dans une application. Lorsque vous utilisez des méthodes issues de ces classes dans votre code, il s'agit de fonctions écrites dans la classe intégrée (similaire à l'exemple de code précédent). Vous pouvez également utiliser les méthodes d'une classe personnalisée que vous créez vous-même.

Les fonctions qui n'appartiennent pas à une classe sont appelées *fonctions de niveau supérieur* (ou encore *fonctions prédéfinies* ou *intégrées*). Cela signifie que vous pouvez les appeler sans constructeur. `trace()` et `setInterval()` sont des exemples de fonctions intégrées au niveau supérieur du langage ActionScript.

Pour ajouter un appel de fonction de niveau supérieur à votre code, il vous suffit d'ajouter une seule ligne de code dans la fenêtre de script du panneau Actions. Par exemple, tapez ce qui suit :

```
trace("my message");
```

Lorsque vous testez le fichier SWF avec cette ligne de code unique, la fonction `trace()` de niveau supérieur est appelée et du texte apparaît dans le panneau de sortie.

N'oubliez pas que lorsque vous souhaitez affecter une méthode à une propriété, vous devez omettre les parenthèses après le nom de la méthode puisque vous transmettez une référence à la fonction :

```
my_mc.myMethod = aFunction;
```

Toutefois, lorsque vous invoquez une méthode dans votre code, vous devez inclure les parenthèses à la suite du nom de la méthode :

```
my_mc.myMethod();
```

REMARQUE

Pour plus d'informations sur les fonctions de niveau supérieur, consultez la section « Présentation des fonctions de niveau supérieur et intégrées », à la page 176.

Vous pouvez également définir des fonctions de nombreuses autres manières. Pour plus d'informations sur chaque type de fonction, consultez les sections suivantes :

- « Présentation des fonctions de niveau supérieur et intégrées », à la page 176
- « Ecriture des fonctions nommées », à la page 177
- « Ecriture de fonctions anonymes et de rappel », à la page 178
- « Présentation des littéraux de fonction », à la page 181
- « Ciblage et appel de fonctions définies par l'utilisateur », à la page 183
- « Présentation des fonctions constructeur », à la page 182

Pour plus d'informations sur l'écriture et l'utilisation des fonctions et méthodes, consultez les sections suivantes. Pour plus d'informations sur l'utilisation des fonctions, consultez la section « Utilisation des fonctions dans Flash », à la page 185. Pour plus d'informations sur l'utilisation des méthodes, consultez la section « Fonctionnement des méthodes », à la page 194.

REMARQUE

Pour plus d'informations sur l'écriture de code à l'aide de l'Assistant de script, consultez le guide *Utilisation de Flash*.

Présentation des fonctions de niveau supérieur et intégrées

Comme l'indique la section « Présentation des fonctions et des méthodes », à la page 171, une fonction est un bloc de code ActionScript qui peut être réutilisé n'importe où dans un fichier SWF. Si vous transmettez des valeurs en tant que paramètres à une fonction, cette dernière agit en conséquence. Une fonction peut également renvoyer des valeurs.

Vous pouvez utiliser les fonctions qui sont intégrées au langage ActionScript. Elles peuvent être de niveau supérieur, comme l'indique la section « Types de méthodes et de fonctions », à la page 173, ou faire partie d'une classe intégrée, telle que Math ou MovieClip, que vous utilisez comme méthode dans votre application.

Dans ActionScript, les fonctions intégrées vous permettent d'effectuer certaines tâches et d'accéder à des informations. Par exemple, à l'aide de la fonction `getTimer()`, vous pouvez obtenir la durée, en millisecondes, de la lecture du fichier SWF. A l'aide de la fonction `getVersion()`, vous pouvez obtenir le numéro de la version du Flash Player qui héberge le fichier. Les fonctions appartenant à un objet sont appelées *méthodes*. Les fonctions qui n'appartiennent pas à un objet sont appelées *fonctions de niveau supérieur* et se trouvent dans les sous-catégories de la catégorie Fonctions globales du panneau Actions.

Certaines fonctions vous obligent à transmettre certaines valeurs. Si vous transmettez plus de paramètres qu'il n'est nécessaire à la fonction, les valeurs supplémentaires sont ignorées. Si vous ne transmettez pas un paramètre nécessaire, les paramètres vides reçoivent le type de données `undefined`, ce qui peut provoquer des erreurs lors de l'exécution du script.

REMARQUE	Pour appeler une fonction, sa définition doit se trouver dans l'image que la tête de lecture a atteinte.

Les fonctions de niveau supérieur sont faciles à utiliser. Pour appeler une fonction, utilisez tout simplement son nom et transmettez les paramètres requis par cette fonction. (Pour plus d'informations sur les paramètres requis, reportez-vous à l'entrée de la fonction dans le *Guide de référence du langage ActionScript 2.0*). Par exemple, ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
trace("my message");
```

Lorsque vous testez le fichier SWF, `my message` apparaît dans le panneau de sortie. Voici deux autres exemples de fonctions de niveau supérieur : `setInterval()` et `getTimer()`. L'exemple qui suit indique comment utiliser ces deux actions ensemble. Ajoutez le code suivant à l'image 1 du scénario :

```
function myTimer():Void {
  trace(getTimer());
}
var intervalID:Number = setInterval(myTimer, 100);
```

Ce code crée un minuteur simple à l'aide de `getTimer()` et utilise les fonctions de niveau supérieur `setInterval()` et `trace()` pour afficher le temps de lecture écoulé (en millisecondes) du fichier SWF dans Flash Player.

L'appel d'une fonction de niveau supérieur est identique à celui d'une *fonction définie par l'utilisateur*. Pour plus d'informations, voir « Ciblage et appel de fonctions définies par l'utilisateur », à la page 183. Pour plus d'informations sur les différentes fonctions, consultez les sections correspondantes du *Guide de référence du langage ActionScript 2.0*.

Ecriture des fonctions nommées

Une fonction nommée est un type de fonction que vous créez généralement dans votre code ActionScript pour effectuer tous types d'actions. Lorsque vous créez un fichier SWF, les fonctions nommées sont d'abord compilées, ce qui signifie que vous pouvez y faire référence à tout endroit de votre code, tant que la fonction a été définie dans l'image en cours ou une image précédente. Par exemple, si une fonction est définie dans l'Image 2 d'un scénario, vous ne pouvez pas accéder à cette fonction dans l'Image 1 du scénario.

Le format standard des fonctions nommées est le suivant :

```
function functionName(parameters) {
  // Bloc de la fonction
}
```

Cet élément de code contient les parties suivantes :

- `functionName` est le nom unique de la fonction. Tous les noms des fonctions d'un document doivent être uniques.
- `parameters` contient un ou plusieurs des paramètres que vous transmettez à la fonction. Les paramètres sont parfois appelés *arguments*. Pour plus d'informations sur les paramètres, consultez la section « Transmission de paramètres à une fonction », à la page 190.
- `// Bloc de la fonction` contient tout le code ActionScript exécuté par la fonction. Cette partie contient les instructions « actives ». Vous pouvez y placer le code à exécuter. Le commentaire `// Bloc de la fonction` est un espace réservé au code correspondant au bloc de la fonction.

Pour utiliser une fonction nommée :

1. Créez un document appelé **namedFunc.fla**.
2. Importez un petit fichier audio dans la bibliothèque. Pour ce faire, cliquez sur Fichier > Importer > Importer dans la bibliothèque, puis sélectionnez un fichier audio.
3. Cliquez sur le fichier audio avec le bouton droit de la souris, puis sélectionnez Liaison.
4. Tapez **mySoundID** dans le champ Identifiant.
5. Sélectionnez l'image 1 du scénario, puis ajoutez le code suivant dans le panneau Actions :

```
function myMessage() {
  trace("mySoundID completed");
}
var my_sound:Sound = new Sound();
my_sound.attachSound("mySoundID");
my_sound.onSoundComplete = myMessage;
my_sound.start();
```

   Dans ce code, vous créez une fonction nommée appelée `myMessage`, que vous utiliserez ensuite dans le script pour appeler une fonction `trace()`.
6. Choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

Pour créer votre propre fonction dans ActionScript, utilisez l'instruction `function`. N'oubliez pas que les paramètres sont facultatifs. Cependant, même sans paramètres, vous devez inclure les parenthèses. Le contenu placé entre accolades (`{}`) est appelé *bloc de fonction*.

Vous pouvez écrire des fonctions dans le scénario principal ou dans des fichiers ActionScript externes, y compris des fichiers de classe.

Vous pouvez également écrire des fonctions constructeur dans des fichiers de classe à l'aide de ce format. Toutefois, dans ce cas, le nom de la fonction correspond à la classe. Pour plus d'informations sur les fonctions constructeur, consultez la section « Ecriture de la fonction constructeur », à la page 244. Reportez-vous également au Chapitre 6, « Classes », à la page 197 pour obtenir des exemples et plus d'informations sur l'écriture de fonctions dans des classes.

Ecriture de fonctions anonymes et de rappel

Une fonction nommée est une fonction à laquelle vous faites référence dans votre script avant ou après l'avoir définie, tandis qu'une *fonction anonyme* est une fonction sans nom qui se référence elle-même. Vous faites référence à la fonction anonyme lors de sa création. En rédigeant votre code ActionScript, vous créerez de nombreuses fonctions anonymes.

Les fonctions anonymes sont fréquemment utilisées avec les gestionnaires d'événements. Pour écrire une fonction anonyme, vous pouvez stocker un littéral de fonction dans une variable. Ainsi, vous pouvez faire référence à la fonction ultérieurement dans votre code. L'exemple suivant montre l'écriture d'une fonction anonyme.

Pour écrire une fonction anonyme :

1. Créez un clip sur la scène, puis sélectionnez-le.
2. Ouvrez l'inspecteur des propriétés et tapez **my_mc** dans la zone de texte Nom de l'occurrence.
3. Sélectionnez l'image 1 du scénario, puis tapez le code suivant dans le panneau Actions :

```
var myWidth = function () {
  trace(my_mc._width);
};
// Plus loin, vous pouvez ajouter dans le code
myWidth();
```

4. Choisissez Contrôle > Tester l'animation.

   La largeur du clip apparaît dans le panneau de sortie.

Vous pouvez également créer une fonction dans un objet, tel que XML ou LoadVars. Vous pouvez associer une fonction anonyme à un événement donné pour créer une *fonction de rappel*. Une fonction appelle une fonction de rappel après un événement spécifique, tel que la fin d'un téléchargement (`onLoad()`) ou la fin d'une animation (`onMotionFinished()`).

Par exemple, vous avez parfois besoin d'écrire du code ActionScript pour manipuler des données chargées dans un fichier SWF à partir du serveur. Après avoir chargé les données dans un fichier SWF, vous pouvez y accéder depuis cet emplacement. Il est important d'utiliser ActionScript pour s'assurer que les données ont bien été entièrement chargées. Vous pouvez utiliser les fonctions de rappel pour envoyer un signal indiquant que les données ont bien été chargées dans le document.

Dans la fonction de rappel suivante, dans laquelle vous chargez un document XML distant, vous associez une fonction anonyme à l'événement `onLoad()`. Vous utilisez `XML.load()` et la fonction de rappel, comme le montre l'exemple suivant. Tapez le code suivant dans l'image 1 du scénario :

```
var my_xml:XML = new XML();
my_xml.onLoad = function(success:Boolean):Void {
  trace(success);
};
my_xml.load("http://www.helpexamples.com/crossdomain.xml");
```

Dans le fragment de code suivant, vous constatez que le gestionnaire d'événement `onLoad()` utilise une fonction anonyme pour gérer l'événement `onLoad()`.

Pour plus d'informations sur les fonctions de rappel, consultez le Chapitre 9, « Gestion d'événements », à la page 311.

Vous pouvez utiliser des fonctions anonymes avec la fonction `setInterval()`, comme le montre le code suivant, qui utilise `setInterval()` pour appeler la fonction anonyme environ toutes les 100 millisecondes (soit toutes les secondes) :

```
setInterval(function() {trace("interval");}, 1000);
```

A la place des fonctions anonymes, vous pouvez utiliser des fonctions nommées. Ces dernières sont souvent plus faciles à interpréter et à comprendre (sauf dans certaines circonstances, par exemple dans le cas de fonctions de rappel). Vous pouvez également pré-référencer une fonction nommée, ce qui signifie que vous y faites référence avant qu'elle n'existe dans le scénario.

Vous ne pouvez pas référencer une fonction anonyme n'importe où dans votre code (sauf si vous l'affectez à une variable), alors que vous pouvez le faire avec une fonction nommée. Supposons par exemple que votre fichier FLA comporte des fonctions anonymes sur l'Image 5, comme indiqué ci-dessous :

```
//avec un clip appelé my_mc qui correspond au scénario
stop();
var myWidth = function () {
  trace(my_mc._width);
};
```

Si vous placez le code suivant sur l'Image 1, il ne peut pas faire référence à la fonction :

```
myWidth();
```

De la même manière, le code suivant placé sur n'importe quelle image ne fonctionne pas :

```
myWidth();
var myWidth:Function = function () {
  trace(my_mc._width);
};
```

En revanche, ce code fonctionne correctement :

```
var myWidth:Function = function () {
  trace(my_mc._width);
};
myWidth();
```

REMARQUE

Vous pouvez également placer `myWidth()` sur n'importe quelle image suivant l'image qui contient la fonction `myWidth`.

Lorsque vous définissez une fonction nommée, vous pouvez l'appeler dans un script d'image même si le code équivalent avec une fonction anonyme ne fonctionne pas :

```
// le code suivant fonctionne car vous appelez une fonction nommée :
myWidth();
function myWidth() {
  trace("foo");
}

// le code suivant ne fonctionne pas car vous appelez une fonction anonyme :
myWidth();
var myWidth:Function = function () {
  trace("foo");
};
```

Pour plus d'informations, voir « Ecriture des fonctions nommées », à la page 177.

REMARQUE

Pour plus d'informations sur l'écriture de code à l'aide de l'Assistant de script, consultez le guide *Utilisation de Flash*.

Présentation des littéraux de fonction

Un *littéral de fonction* est une fonction sans nom que vous déclarez dans une expression et non dans une instruction. Ces fonctions littérales s'avèrent très pratiques lorsque vous devez utiliser une fonction momentanément ou une fonction dans votre code à la place d'une expression. La syntaxe des littéraux de fonction est la suivante :

```
function (param1, param2, etc) {
  // instructions
};
```

Par exemple, le code suivant utilise un littéral de fonction en tant qu'expression :

```
var yourName:String = "Ester";
setInterval(function() {trace(yourName);}, 200);
```

REMARQUE

Lorsque vous redéfinissez un littéral de fonction, la nouvelle définition de fonction remplace l'ancienne.

Vous pouvez stocker un littéral de fonction dans une variable afin d'y accéder ultérieurement dans votre code. Pour ce faire, vous utilisez une *fonction anonyme*. Pour plus d'informations, voir « Ecriture de fonctions anonymes et de rappel », à la page 178.

Présentation des fonctions constructeur

Le constructeur d'une classe est une fonction spéciale appelée automatiquement lorsque vous créez une occurrence de classe en utilisant le mot-clé `new` (tel que `var my_xml:XML = new XML();`). La fonction constructeur porte le même nom que la classe qui la contient. Par exemple, une classe Person personnalisée que vous créez contiendrait la fonction constructeur suivante :

```
public function Person(speed:Number) {
  Person.numPeople++;
  this._speed = speed;
}
```

Vous pouvez alors créer une nouvelle occurrence à l'aide de :

```
var myPerson:Person = new Person();
```

REMARQUE	Si vous ne déclarez aucune fonction constructeur explicitement dans votre fichier de classe, c'est-à-dire, si vous ne créez pas de fonction dont le nom correspond à celui de la classe, le compilateur crée automatiquement une fonction constructeur vide.

Une classe ne peut contenir qu'une seule fonction constructeur ; les fonctions constructeur étendues ne sont pas autorisées dans ActionScript 2.0. En outre, une fonction constructeur ne peut pas avoir de type de renvoi. Pour plus d'informations sur l'écriture de fonctions constructeur dans des fichiers de classe, consultez la section « Ecriture de la fonction constructeur », à la page 244.

Définition des fonctions globales et de scénario

Dans la section « Présentation des fonctions et des méthodes », à la page 171, vous avez découvert les différentes sortes de fonctions disponibles dans Flash. Comme les variables, les fonctions sont associées au scénario du clip qui les définit, et vous devez utiliser un chemin cible pour les appeler. Comme pour les variables, vous pouvez utiliser l'identifiant `_global` pour déclarer une fonction globale accessible à tous les scénarios et les domaines sans employer de chemin cible. Pour définir une fonction globale, faites précéder son nom de l'identifiant `_global`, comme illustré ci-dessous :

```
_global.myFunction = function(myNum:Number):Number {
   return (myNum * 2) + 3;
};
trace(myFunction(5)) // 13
```

Pour plus d'informations sur `_global` et le domaine, consultez la section « Variables et domaine », à la page 62.

Pour définir une fonction de scénario, utilisez l'instruction `function` suivie du nom de la fonction, des paramètres qui doivent être transmis à la fonction et des instructions ActionScript qui indiquent le rôle de la fonction.

L'exemple suivant illustre une fonction nommée `areaOfCircle` et dotée du paramètre `radius` :

```
function areaOfCircle(radius:Number):Number {
  return (Math.PI * radius * radius);
}
trace (areaOfCircle(8));
```

Vous pouvez également définir des fonctions de nombreuses autres manières. Pour plus d'informations sur chaque type de fonction, consultez les sections suivantes :

- « Présentation des fonctions de niveau supérieur et intégrées », à la page 176
- « Ecriture des fonctions nommées », à la page 177
- « Ecriture de fonctions anonymes et de rappel », à la page 178
- « Présentation des littéraux de fonction », à la page 181
- « Présentation des fonctions constructeur », à la page 182
- « Ciblage et appel de fonctions définies par l'utilisateur », à la page 183

Pour plus d'informations sur l'appellation des fonctions, consultez la section « Appellation des fonctions », à la page 185. Pour des exemples concrets d'emploi de fonctions dans un fichier de classe externe, consultez la section « Utilisation des fonctions dans Flash », à la page 185 et le Chapitre 6, « Classes », à la page 197.

REMARQUE	Pour plus d'informations sur l'écriture de code à l'aide de l'Assistant de script, consultez le guide *Utilisation de Flash*.

Ciblage et appel de fonctions définies par l'utilisateur

Les *fonctions définies par l'utilisateur* sont celles que vous créez vous-même pour les utiliser dans vos applications, à la différence des fonctions des classes intégrées qui exécutent des fonctions prédéfinies. Vous choisissez leur nom et ajoutez des instructions dans le bloc de la fonction. Les sections précédentes traitent de l'écriture de différentes fonctions (fonctions de rappel, nommées et sans nom). Pour plus d'informations sur l'appellation des fonctions, consultez la section « Appellation des fonctions », à la page 185. Pour l'utilisation des fonctions, consultez la section « Utilisation des fonctions dans Flash », à la page 185.

Vous pouvez utiliser un chemin cible pour appeler une fonction d'un scénario, quel qu'il soit, à partir de n'importe quel autre scénario, y compris celui d'un fichier SWF chargé.
Pour appeler une fonction, tapez le chemin cible dans son nom, si nécessaire, puis transmettez tous les paramètres requis entre parenthèses. Il existe plusieurs formes de syntaxe pour les fonctions définies par l'utilisateur. Le code suivant utilise un chemin pour appeler la fonction `initialize()` qui a été définie dans le scénario actuel et n'exige aucun paramètre :

```
this.initialize();
```

L'exemple suivant utilise un chemin relatif pour appeler la fonction `list()` qui a été définie dans le clip `functionsClip` :

```
this._parent.functionsClip.list(6);
```

Pour plus d'informations sur l'écriture des fonctions nommées, consultez la section « Ecriture des fonctions nommées », à la page 177. Pour plus d'informations sur les paramètres, consultez la section « Transmission de paramètres à une fonction », à la page 190.

Vous pouvez également définir vos propres fonctions nommées. Par exemple, la fonction nommée suivante `helloWorld()` est définie par l'utilisateur :

```
function helloWorld() {
  trace("Hello world!");
};
```

L'exemple suivant montre comment utiliser une fonction définie par l'utilisateur dans un fichier FLA.

Pour créer et appeler une simple fonction définie par l'utilisateur :

1. Créez un document Flash, puis enregistrez-le sous le nom **udf.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

   ```
   function traceHello(name:String):Void {
     trace("hello, " + name + "!");
   }
   traceHello("world"); // hello, world!
   ```

   Le code précédent crée une fonction définie par l'utilisateur appelée `traceHello()` qui prend un argument, `name`, et suit un message de bienvenue. Pour appeler la fonction définie par l'utilisateur, vous pouvez appeler `traceHello` à partir du même scénario que la définition de fonction et transmettre une seule valeur de chaîne.
3. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

Pour plus d'informations sur les fonctions nommées, consultez la section « Ecriture des fonctions nommées », à la page 177. Les classes contiennent de nombreuses fonctions définies par l'utilisateur. Pour plus d'informations sur l'écriture des fonctions dans des fichiers de classe, consultez la section « Utilisation des fonctions dans Flash », à la page 185. Reportez-vous également aux sections suivantes dans le Chapitre 6, « Classes » : « Utilisation des méthodes et des propriétés d'un fichier de classe », à la page 219, « Présentation des méthodes et propriétés (membres) publiques, privées et statiques », à la page 221 et « Présentation des membres de classe », à la page 224.

Appellation des fonctions

Les noms de fonction doivent commencer par une minuscule. Ils doivent décrire la valeur renvoyée par la fonction, le cas échéant. Par exemple, si la fonction renvoie le titre d'une chanson, nommez-la `getCurrentSong()`.

Etablissez des normes d'appellation pour regrouper les fonctions similaires (celles qui sont apparentées par leur fonctionnalité), car ActionScript n'autorise pas la surcharge. Dans le cas de la programmation orientée objet (OOP), la *surcharge* consiste à autoriser des comportements différents pour vos fonctions, selon les types de données qui leurs sont passés.

Comme avec les variables, vous ne pouvez pas utiliser de caractères spéciaux et le nom de méthode ne peut pas débuter par un chiffre. Pour plus d'informations, voir « Conventions d'appellation », à la page 717. Pour plus d'informations sur l'appellation des méthodes, consultez la section « Appellation des méthodes », à la page 196.

Utilisation des fonctions dans Flash

Cette section présente l'utilisation des fonctions dans une application. Certains des exemples de code suivants utilisent du code ActionScript qui réside dans le fichier FLA. Les autres placent les fonctions dans un fichier de classe à titre de comparaison. Pour obtenir plus d'informations et des exemples sur l'utilisation des fonctions dans un fichier de classe, consultez le Chapitre 6, « Classes », à la page 197. Pour obtenir des informations détaillées et des instructions sur la façon d'écrire des fonctions pour un fichier de classe, consultez la section « Exemple : Ecriture de classes personnalisées », à la page 238.

Pour réduire la somme de travail nécessaire, ainsi que la taille de votre fichier SWF, réutilisez des blocs de code existants chaque fois que possible. Pour ce faire, vous pouvez appeler la même fonction à plusieurs reprises, au lieu de créer du code de toutes pièces à chaque fois. Les fonctions peuvent comporter du code générique, ce qui permet d'utiliser les mêmes blocs à des fins légèrement différentes dans un fichier SWF. Le recyclage du code permet de créer des applications efficaces et de réduire la quantité de code ActionScript à écrire, et donc la durée du développement.

Vous pouvez créer des fonctions dans un fichier FLA ou de classe ou écrire du code ActionScript qui réside dans un composant à base de code. Les exemples suivants présentent la création de fonctions sur un scénario et dans un fichier de classe.

CONSEIL	En réunissant vos scripts dans des fichiers de classe ou des composants à base de code, vous pouvez facilement partager, distribuer et réutiliser vos blocs de code. L'utilisateur peut installer votre composant, le faire glisser sur la scène et utiliser le code que vous avez stocké dans le fichier, tel que le flux de travail pour les composants à base de code disponibles dans (Fenêtre › Bibliothèques communes › Classes).

L'exemple suivant montre comment créer et appeler une fonction dans un fichier FLA.

Pour créer et appeler une fonction dans un fichier FLA :

1. Créez un document Flash, puis enregistrez-le sous le nom **basicFunction.fla**.
2. Choisissez Fenêtre > Actions pour ouvrir le panneau Actions.
3. Tapez le code ActionScript suivant dans la fenêtre de script :

   ```
   function helloWorld() {
     // instructions ici
     trace("Hello world!");
   };
   ```

   Ce code ActionScript définit la fonction (nommée et définie par l'utilisateur) appelée `helloWorld()`. Si vous testez votre fichier SWF à ce stade, rien ne se produit.
   Par exemple, vous ne voyez pas l'instruction `trace` dans le panneau de sortie. Pour ce faire, vous devez appeler la fonction `helloWorld()`.
4. Tapez la ligne suivante à la suite de la fonction :

   ```
   helloWorld();
   ```

   Ce code appelle la fonction `helloWorld()`.
5. Choisissez Contrôle > Tester l'animation pour tester le fichier FLA.

   Le texte suivant apparaît dans le panneau de sortie : `Hello world!`

Pour plus d'informations sur la transmission de valeurs (paramètres) à une fonction, consultez la section « Transmission de paramètres à une fonction », à la page 190.

Il existe plusieurs manières pour écrire des fonctions dans le scénario principal. Les plus utilisées sont les fonctions nommées et anonymes. Par exemple, vous pouvez utiliser la syntaxe suivante pour créer des fonctions :

```
function myCircle(radius:Number):Number {
  return (Math.PI * radius * radius);
}
trace(myCircle(5));
```

Les fonctions anonymes sont souvent plus difficiles à interpréter. Comparez le code suivant au précédent.

```
var myCircle:Function = function(radius:Number):Number {
  // Bloc de fonction ici
  return (Math.PI * radius * radius);
};
trace(myCircle(5));
```

Vous pouvez également placer des fonctions dans des fichiers de classe si vous utilisez ActionScript 2.0, comme l'illustre l'exemple suivant :

```
class Cercle {
  public function area(radius:Number):Number {
    return (Math.PI * Math.pow(radius, 2));
  }
  public function perimeter(radius:Number):Number {
    return (2 * Math.PI * radius);
  }
  public function diameter(radius:Number):Number {
    return (radius * 2);
  }
}
```

Pour plus d'informations sur l'écriture de fonctions dans un fichier de classe, consultez le Chapitre 6, « Classes », à la page 197.

Comme l'indique l'exemple de code précédent, vous n'avez pas besoin de placer des fonctions sur un scénario. L'exemple suivant place également des fonctions dans un fichier de classe. Cette technique est judicieuse lorsque vous créez des applications volumineuses à l'aide d'ActionScript 2.0 car elle vous permet de réutiliser facilement votre code dans plusieurs applications. Lorsque vous devez utiliser de nouveau les fonctions dans d'autres applications, vous pouvez importer la classe existante sans avoir à ré-écrire le code de toutes pièces ou dupliquer les fonctions dans la nouvelle application.

Pour créer des fonctions dans un fichier de classe :

1. Créez un document ActionScript, puis enregistrez-le sous le nom **Utils.fla**.
2. Tapez le code ActionScript suivant dans la fenêtre de script :

```
class Utils {
  public static function randomRange(min:Number, max:Number):Number {
    if (min > max) {
      var temp:Number = min;
      min = max;
      max = temp;
    }
    return (Math.floor(Math.random() * (max - min + 1)) + min);
  }
  public static function arrayMin(num_array:Array):Number {
    if (num_array.length == 0) {
      return Number.NaN;
    }
    num_array.sort(Array.NUMERIC | Array.DESCENDING);
    var min:Number = Number(num_array.pop());
    return min;
  }
```

```
  public static function arrayMax(num_array:Array):Number {
    if (num_array.length == 0) {
      return undefined;
    }
    num_array.sort(Array.NUMERIC);
    var max:Number = Number(num_array.pop());
    return max;
  }
}
```

3. Sélectionnez Fichier > Enregistrer pour enregistrer le fichier ActionScript.
4. Créez un document Flash et enregistrez-le sous le nom **classFunctions.fla** dans le même répertoire que Utils.as.
5. Choisissez Fenêtre > Actions pour ouvrir le panneau Actions.
6. Tapez le code ActionScript suivant dans la fenêtre de script :

```
var randomMonth:Number = Utils.randomRange(0, 11);
var min:Number = Utils.arrayMin([3, 3, 5, 34, 2, 1, 1, -3]);
var max:Number = Utils.arrayMax([3, 3, 5, 34, 2, 1, 1, -3]);
trace("month: " + randomMonth);
trace("min: " + min); // -3
trace("max: " + max); // 34
```

7. Choisissez Contrôle > Tester l'animation pour tester les documents. Le texte suivant apparaît dans le panneau de sortie :

```
month: 7
min: -3
max: 34
```

REMARQUE

Pour plus d'informations sur l'écriture de code à l'aide de l'Assistant de script, consultez le guide *Utilisation de Flash*.

Utilisation de variables dans les fonctions

Les variables locales sont des outils qui simplifient grandement l'organisation du code et en facilitent la compréhension. Lorsqu'une fonction utilise des variables locales, elle peut les cacher à tous les autres scripts du fichier SWF. Les variables locales ont un domaine limité au corps de la fonction et sont détruites à la sortie de celle-ci. Flash traite également tous les paramètres transmis à une fonction en tant que variables locales.

REMARQUE	Vous pouvez également utiliser des variables normales dans une fonction. Cependant, si vous modifiez des variables normales dans une fonction, il est judicieux de commenter ces changements.

Pour utiliser des variables dans les fonctions :

1. Créez un document Flash, puis enregistrez-le sous le nom **flashvariables.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
var myName:String = "Ester";
var myAge:String = "65";
var myFavSoftware:String = "Flash";
function traceMe(yourFavSoftware:String, yourName:String,
  yourAge:String) {
  trace("I'm " + yourName + ", I like " + yourFavSoftware + ", and I'm "
  + yourAge + ".");
}
traceMe(myFavSoftware, myName, myAge);
```

3. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

Pour plus d'informations sur la transmission de paramètres, consultez la section « Transmission de paramètres à une fonction », à la page 190. Pour plus d'informations sur les variables et les données, consultez le Chapitre 3, « Données et types de données », à la page 35.

Transmission de paramètres à une fonction

Les paramètres, également appelés *arguments*, sont les éléments sur lesquels une fonction exécute son code. Dans cet ouvrage, les termes *paramètre* et *argument* sont interchangeables. Vous pouvez transmettre des paramètres (valeurs) à une fonction. Vous pouvez ensuite utiliser ces paramètres pour le traitement de la fonction. Vous utilisez les valeurs au sein du bloc de la fonction (ses instructions).

Certains paramètres sont obligatoires, d'autres sont facultatifs. Vous pouvez même avoir des paramètres obligatoires et facultatifs dans la même fonction. Si vous ne transmettez pas suffisamment de paramètres à une fonction, Flash définit les valeurs des paramètres manquants sur `undefined`, ce qui peut générer des résultats inattendus dans votre fichier SWF.

La fonction suivante, appelée `myFunc()`, prend le paramètre `someText` :

```
function myFunc(someText:String):Void {
  trace(someText);
}
```

Après la transmission du paramètre, vous pouvez transmettre une valeur à la fonction lorsque vous appelez celle-ci. Cette valeur apparaît dans le panneau de sortie, comme suit :

```
myFunc("This is what traces");
```

Lorsque vous appelez la fonction, vous devez toujours transmettre le nombre de paramètres spécifié sauf si votre fonction recherche des valeurs indéfinies et si les valeurs par défaut sont définies en conséquence. La fonction substitue les valeurs transmises aux paramètres de la définition de la fonction. Si l'un des paramètres est manquant, Flash lui affecte la valeur `undefined`. Dans votre code ActionScript, vous êtes fréquemment amené(e) à transmettre des paramètres aux fonctions.

Vous pouvez également transmettre plusieurs paramètres à la même fonction, aussi simplement que dans l'exemple suivant :

```
var birthday:Date = new Date(1901, 2, 3);
trace(birthday);
```

Les paramètres sont séparés par des virgules. De nombreuses fonctions intégrées du langage ActionScript possèdent plusieurs paramètres. Par exemple, la méthode `startDrag()` de la classe MovieClip prend cinq paramètres : `lockCenter`, `left`, `top`, `right` et `bottom`.

```
startDrag(lockCenter:Boolean, left:Number, top:Number, right:Number,
  bottom:Number):Void
```

Pour transmettre un paramètre à une fonction :

1. Créez un document Flash, puis enregistrez-le sous le nom **parameters.fla**.
2. Ajoutez le code suivant à l'image 1 du scénario :

```
function traceMe(yourMessage:String):Void {
  trace(yourMessage);
}
traceMe("Comment allez-vous ?");
```

   Les premières lignes du code créent une fonction définie par l'utilisateur, appelée `traceMe()`, qui prend un seul paramètre, `yourMessage`. La dernière ligne du code appelle la fonction `traceMe()` et transmet la valeur de chaîne « `Comment allez-vous ?` ».
3. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

L'exemple suivant montre comment transmettre plusieurs paramètres à une fonction.

Pour transmettre plusieurs paramètres à une fonction :

1. Créez un document Flash, puis enregistrez-le sous le nom **functionTest.fla**.
2. Ajoutez le code suivant à l'image 1 du scénario principal :

```
function getArea(width:Number, height:Number):Number {
     return width * height;
}
```

   La fonction `getArea()` réclame deux paramètres : `width` et `height`.
3. Tapez le code suivant à la suite de la fonction :

```
var area:Number = getArea(10, 12);
trace(area); // 120
```

   L'appel de la fonction `getArea()` affecte les valeurs 10 et 12 aux paramètres width et height, respectivement, et vous enregistrez la valeur renvoyée dans l'occurrence `area`. Ensuite, vous suivez les valeurs enregistrées dans l'occurrence `area`.
4. Choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

   La mention `120` s'affiche dans le panneau de sortie.

   Les paramètres de la fonction `getArea()` sont similaires aux valeurs d'une variable locale : ils existent tant que la fonction est appelée et cessent d'exister à la sortie de la fonction.

Dans l'exemple suivant, le code ActionScript renvoie la valeur `NaN` (Non numérique) si vous ne renseignez pas suffisamment de paramètres dans la fonction `addNumbers()`.

Pour transmettre un nombre variable de paramètres à une fonction :

1. Créez un document Flash, puis enregistrez-le sous le nom **functionTest2.fla**.
2. Ajoutez le code suivant à l'image 1 du scénario principal :

```
function addNumbers(a:Number, b:Number, c:Number):Number {
  return (a + b + c);
}
trace(addNumbers(1, 4, 6)); // 11
trace(addNumbers(1, 4)); // NaN (Non numérique), c est undefined
trace(addNumbers(1, 4, 6, 8)); // 11
```

   Si vous ne transmettez pas suffisamment de paramètres à la fonction `addNumbers`, la valeur `undefined` est affectée par défaut aux arguments manquants. Si vous en transmettez trop, les arguments excédentaires sont ignorés.
3. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

   Flash affiche alors les valeurs suivantes : 11, NaN, 11.

Renvoi de valeurs depuis les fonctions

Utilisez l'instruction `return` pour renvoyer des valeurs depuis les fonctions. L'instruction `return` indique la valeur renvoyée par une fonction. L'instruction `return` renvoie le résultat d'une évaluation sous la forme d'une valeur de la fonction dans laquelle une expression est exécutée. L'instruction `return` renvoie son résultat immédiatement au code appelant.

Pour plus d'informations, consultez la section `instruction return` du *Guide de référence du langage ActionScript 2.0*.

Les règles suivantes régissent l'emploi de l'instruction `return` dans les fonctions :

- Si vous spécifiez un type de renvoi autre que Void pour une fonction, vous devez inclure une instruction `return` suivie par la valeur renvoyée dans la fonction.
- Si vous spécifiez un type de renvoi Void, l'instruction `return` est inutile, mais dans ce cas, elle ne doit être suivie par aucune valeur.
- Quel que soit le type de renvoi choisi, vous pouvez utiliser une instruction `return` pour quitter la fonction en son milieu.
- Si vous ne spécifiez pas de type de renvoi, l'incorporation d'une instruction `return` est facultative.

Par exemple, la fonction suivante renvoie le carré du paramètre `myNum` et spécifie que la valeur renvoyée doit être de type Number :

```
function sqr(myNum:Number):Number {
  return myNum * myNum;
}
```

Certaines fonctions effectuent une série de tâches sans renvoyer de valeur. L'exemple suivant renvoie la valeur traitée. Vous récupérez cette valeur dans une variable, puis vous utilisez cette variable dans votre application.

Pour renvoyer une valeur et la récupérer dans une variable :

1. Créez un document Flash, puis enregistrez-le sous le nom **return.fla**.
2. Ajoutez le code suivant à l'image 1 du scénario principal :

```
function getArea(width:Number, height:Number):Number {
     return width * height;
}
```

   La fonction `getArea()` réclame deux paramètres : `width` et `height`.

3. Tapez le code suivant à la suite de la fonction :

```
var area:Number = getArea(10, 12);
trace(area); // 120
```

   L'appel de la fonction `getArea()` affecte les valeurs 10 et 12 aux paramètres width et height respectivement et vous enregistrez la valeur renvoyée dans l'occurrence `area`. Ensuite, vous suivez les valeurs enregistrées dans l'occurrence `area`.

4. Choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

   La mention `120` s'affiche dans le panneau de sortie.

   Les paramètres de la fonction `getArea()` sont similaires aux valeurs d'une variable locale : ils existent tant que la fonction est appelée et cessent d'exister à la sortie de la fonction.

Présentation des fonctions imbriquées

Vous pouvez appeler une fonction de l'intérieur d'une autre fonction. Ainsi, vous pouvez imbriquer des fonctions les unes dans les autres pour qu'elles exécutent des tâches spéciales dans Flash.

Par exemple, vous pouvez imbriquer plusieurs fonctions dans un scénario pour exécuter des tâches spécifiques sur une chaîne. Tapez le code suivant dans l'image 1 du scénario :

```
var myStr:String = "My marshmallow chicken is yellow.";
trace("Original string: " + myStr);
function formatText():Void {
  changeString("Put chicken in microwave.");
  trace("Changed string: " + myStr);
}
function changeString(newtext:String):Void {
  myStr = newtext;
}
// Appel de la fonction.
formatText();
```

Choisissez Contrôle > Tester l'animation pour tester la fonction imbriquée. Les deux fonctions `formatText()` et `changeString()` sont appliquées à la chaîne lorsque vous appelez la fonction `formatText()`.

Fonctionnement des méthodes

Les méthodes sont des fonctions associées à une classe. La classe peut être personnalisée ou intégrée au langage ActionScript. Pour en savoir plus sur la comparaison entre les méthodes et les fonctions, consultez les sections « Présentation des fonctions et des méthodes », à la page 171 et « Types de méthodes et de fonctions », à la page 173.

Par exemple, `sortOn()` est une méthode intégrée associée à la classe Array (`sortOn` est une fonction de la classe Array prédéfinie intégrée à Flash).

Pour utiliser la méthode sortOn() dans un fichier FLA :

1. Créez un document Flash et enregistrez-le sous le nom **methods.fla**.
2. Ajoutez le code suivant à l'image 1 du scénario :

```
var userArr:Array = new Array();
userArr.push({firstname:"George", age:39});
userArr.push({firstname:"Dan", age:43});
userArr.push({firstname:"Socks", age:2});
userArr.sortOn("firstname");
var userArrayLenth:Number = userArr.length;
var i:Number;
for (i = 0; i < userArrayLenth; i++) {
  trace(userArr[i].firstname);
}
```

   Vous utilisez la méthode `sortOn()` de la classe Array pour créer un nouvel objet Array appelé `userArr`. Ce nouveau tableau est rempli par trois objets contenant un prénom et un âge, puis il est trié sur la base de la valeur de la propriété `firstname` de chaque objet. Pour finir, vous passez en boucle sur chaque élément du tableau, vous affichez le prénom dans le panneau de sortie et vous triez les noms par ordre alphabétique.
3. Choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

   Ce code donne les résultats suivants dans le panneau de sortie :

```
Dan
George
Socks
```

Comme le démontre la section « Ecriture des fonctions nommées », à la page 177, lorsque vous écrivez le code suivant sur l'image 1 du scénario, votre code ActionScript définit une fonction appelée `eatCabbage()`.

```
function eatCabbage() {
    trace("tastes bad");
}
eatCabbage();
```

Toutefois, si vous écrivez la fonction `eatCabbage()` dans un fichier de classe et que vous appelez, par exemple, la fonction `eatCabbage()` dans le fichier FLA, `eatCabbage()` est considérée comme une méthode.

Les exemples suivants montrent comment créer des méthodes dans une classe.

Pour comparer des méthodes et des fonctions :

1. Créez un fichier ActionScript, sélectionnez Fichier > Enregistrer sous, puis enregistrez-le sous le nom **EatingHabits.as**.
2. Tapez le code ActionScript suivant dans la fenêtre de script :

```
class EatingHabits {
  public function eatCabbage():Void {
    trace("tastes bad");
  }
}
```

3. Enregistrez les modifications apportées au fichier EatingHabits.as.
4. Créez un document Flash, sélectionnez Fichier > Enregistrer sous, nommez-le **methodTest.fla**, puis enregistrez ce fichier dans le même répertoire que EatingHabits.as.
5. Tapez le code ActionScript suivant dans l'image 1 du scénario :

```
var myHabits:EatingHabits = new EatingHabits();
myHabits.eatCabbage();
```

   Lorsque vous utilisez ce code ActionScript, vous appelez la méthode `eatCabbage()` de la classe EatingHabits.

REMARQUE	Lorsque vous utilisez des méthodes d'une classe intégrée (en plus de la classe personnalisée que vous avez créée précédemment dans cette procédure), vous utilisez une *méthode* sur un scénario.

6. A la suite de la ligne ActionScript précédente, ajoutez le code suivant :

```
function eatCarrots():Void {
    trace("tastes good");
}
eatCarrots();
```

   Dans ce code, vous écrivez et appelez la fonction `eatCarrots()`.
7. Choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

Appellation des méthodes

Utilisez de préférence des verbes pour nommer les méthodes et des casses différentes pour les mots concaténés, en conservant la première lettre en minuscule. Par exemple, vous pouvez nommer les méthodes de la façon suivante :

```
chanter();
boogie();
chanterFort();
danserVite();
```

Les verbes sont généralement associés aux noms des méthodes dans la mesure où ces dernières exécutent une opération sur un objet. Comme avec les variables, vous ne pouvez pas utiliser de caractères spéciaux et le nom de méthode ne peut pas débuter par un chiffre. Pour plus d'informations, voir « Conventions d'appellation », à la page 717.

CHAPITRE 6

Classes

6

Ce chapitre présente l'utilisation et l'écriture des classes avec ActionScript 2.0. Les classes sont le cœur d'ActionScript 2.0 et leur importance s'est encore accrue dans cette version de Flash. Vous vérifierez par vous-même la véracité de cette assertion tout au long de ce chapitre.

Ce chapitre commence par présenter la terminologie de base et ses rapports avec les classes et la programmation orientée objet (OOP). Vous découvrez ensuite un exemple de fichier de classe, avec le fonctionnement de chacune de ses sections et l'organisation de la classe. Le reste du chapitre vous guidera dans la création de vos propres classes personnalisées et dans leur utilisation au sein de vos documents Flash. Vous découvrez les chemins de classe de Flash et apprenez à documenter vos classes de sorte que les personnes amenées à lire ou utiliser votre code puissent aisément le comprendre, ainsi que l'objectif général de vos classes.

Cette section contient les exemples de code qui permettent de se familiariser avec la création de classes dans ActionScript 2.0. A la fin de ce chapitre, vous devriez pouvoir écrire un fichier de classe typique, comprendre et reconnaître les classes de Flash et être à même de lire sans peine les fichiers de classes créés par d'autres personnes.

Si vous ne maîtrisez pas la rédaction de scripts ActionScript 2.0, consultez le Chapitre 17, « Recommandations et conventions de programmation pour ActionScript 2.0 », à la page 715 et le Chapitre 4, « Eléments fondamentaux du langage et de la syntaxe », à la page 81.

Pour plus d'informations sur l'utilisation des classes intégrées et personnalisée de données, consultez les rubriques suivantes :

Présentation de la programmation orientée objet et de Flash

ActionScript 2.0 est un langage orienté objet. Ces langages reposent sur le concept de *classes* et d'*occurrences*. Une classe définit toutes les propriétés qui distinguent une série d'objets. Par exemple, une classe User représente un groupe d'utilisateurs qui utilisent votre application. Puis, vous avez une *instanciation* de la classe, laquelle, pour la classe User, correspond à l'un des utilisateurs individuels—l'un des membres de la classe. L'instanciation produit une *occurrence* de la classe User qui présente toutes les propriétés de la classe User.

Les classes sont également considérées comme des *types de données* ou des *modèles*, que vous pouvez créer pour définir un nouveau type d'objet. Par exemple, si vous avez besoin du type de données Lettuce dans votre application, vous pouvez écrire la classe Lettuce. L'opération définit l'objet Lettuce et vous permet d'affecter ensuite vos méthodes Lettuce (`wash()`) et vos propriétés (`leafy` ou `bugs`). Pour définir une classe, vous utilisez le mot-clé `class` dans un fichier de script externe. Pour créer ce fichier de script externe dans l'outil de programmation de Flash, choisissez Fichier > Nouveau puis sélectionnez Fichier ActionScript.

ActionScript 2.0 inclut des fonctionnalités telles que des effets de filtre, le chargement et le téléchargement de fichiers et l'API externe ; il reprend également plusieurs concepts et mots-clés de la programmation orientée objet (tels que `class`, `interface` et `package`) provenant d'autres langages, tels que Java. Le langage de programmation vous permet de créer des structures logicielles réutilisables, évolutives, fiables et faciles à maintenir. En fournissant aux utilisateurs une aide à la programmation et des informations de débogage approfondies, il réduit également les délais de développement. ActionScript2.0 permet de créer des objets et d'établir des héritages, de créer des classes personnalisées et d'étendre les classes intégrées et de niveau supérieur de Flash. Dans ce chapitre, vous allez apprendre à créer des classes personnalisées et à les utiliser.

Flash comprend près de 65 classes intégrées et de niveau supérieur couvrant à peu près tous les types de données, de base ou primitifs (Array, Boolean, Date, etc.), jusqu'aux erreurs et événements personnalisés, ainsi que plusieurs moyens de charger du contenu externe (XML, images, données binaires brutes, etc.). Vous pouvez également écrire vos propres classes personnalisées et les intégrer dans vos documents Flash, ou encore étendre les classes de niveau supérieur et ajouter vos propres fonctionnalités ou modifier celles qui existent. Par exemple « Présentation des membres de classe », à la page 224 dans ce chapitre, explique comment créer une classe personnalisée qui contient des propriétés personnalisées pour le nom et l'âge des individus. Vous pouvez ensuite traiter cette classe personnalisée comme un nouveau type de données dans vos documents et en créer une nouvelle occurrence à l'aide du `nouvel`opérateur.

Pour plus d'informations sur l'utilisation des OOP, consultez les rubriques suivantes :

- « Avantages de l'utilisation des classes », à la page 200
- « Présentation des packages », à la page 200
- « Présentation des valeurs et des types de données », à la page 204
- « Bases de la programmation orientée objet. », à la page 204

Avantages de l'utilisation des classes

En programmation orientée objet, une classe définit une catégorie d'objets. Une classe décrit les propriétés (données) et les méthodes (comportements) d'un objet, comme un plan d'architecte décrit les caractéristiques d'un immeuble. Vous écrivez votre classe personnalisée dans un fichier ActionScript (AS) externe et pouvez ensuite l'importer dans votre application lorsque vous compilez le fichier FLA.

Les classes se révèlent très utiles lors du développement d'applications Flash volumineuses car elles permettent d'organiser une grande partie de la complexité du code dans des fichiers de classes externes. Lorsque vous déplacez une bonne partie du code vers une classe personnalisée, vous simplifiez non seulement la réutilisation ultérieure du code, mais vous pouvez également « masquer » certaines des méthodes et propriétés des autres parties du code ActionScript. Vous évitez ainsi que les utilisateurs accèdent à des informations confidentielles ou puissent modifier des données qui ne doivent pas l'être.

En utilisant une classe, vous pouvez également étendre les classes existantes et ajouter de nouvelles fonctionnalités ou modifier celles qui existent. Par exemple, si vous créez trois classes très similaires, vous pouvez écrire une classe de base, puis deux autres classes qui sont une extension de la première. Ces deux classes peuvent ajouter des méthodes et des propriétés supplémentaires, ainsi il n'est pas nécessaire de créer trois fichiers de classe dupliquant tous le même code et la même logique.

La possibilité de réutilisation du code est un autre avantage des classes. Par exemple, si vous créez une classe personnalisée qui crée une barre de progression personnalisée via l'interface de programmation d'applications de dessin (API), vous pouvez enregistrer la classe de la barre de progression dans votre chemin de classe, puis réutiliser le même code dans tous vos documents Flash en important la classe personnalisée Pour plus d'informations sur la configuration du chemin de classe, consultez « Importation de fichiers de classe », à la page 212 et « Définition et modification du chemin de classe », à la page 214.

Présentation des packages

Lorsque vous créez des classes, placez vos fichiers de classe ActionScript dans des *packages*. Un package est un répertoire qui contient un ou plusieurs fichiers de classe et qui réside dans un répertoire de chemin de classe désigné (consultez les sections « Importation de fichiers de classe », à la page 212 et « Définition et modification du chemin de classe », à la page 214). Un package peut également contenir d'autres packages, appelés *sous-packages*, chacun possédant ses propres fichiers de classe.

Comme les variables, les noms de packages doivent être des identifiants. Ainsi, le premier caractère peut être une lettre, un caractère de soulignement (_) ou le signe dollar ($), et chaque caractère suivant doit être une lettre, un nombre, un caractère de soulignement ou le signe dollar. Il y a des façons conseillées de nommer des packages, par exemple en évitant d'utiliser des caractères de soulignement ou le signe dollar. Pour plus d'informations sur les appellations des packages, consultez le « Appellation des packages », à la page 726..

Les packages sont généralement utilisés pour organiser des classes connexes. Vous pouvez par exemple avoir trois classes connexes, Carre, Cercle et Triangle, définies dans Carre.as, Cercle.as et Triangle.as. Supposons que vous ayez enregistré les fichiers ActionScript dans un répertoire spécifié dans le chemin de classe, comme dans l'exemple suivant :

```
// Dans Carre.as :
class Square {}

// Dans Cercle.as :
class Circle {}

// Dans Triangle.as :
class Triangle {}
```

Etant donné que ces trois classes sont connexes, vous pouvez choisir de les placer dans un package (répertoire) nommé Formes. Dans ce cas, le nom complet de la classe contient le chemin du package aussi bien que le nom de classe simple. Les chemins de package sont symbolisés par une syntaxe de point (.), chaque point représentant un sous-répertoire.

Par exemple, si vous placez tous les fichiers ActionScript qui définissent une forme dans le répertoire Formes, vous devrez alors changer le nom de chaque fichier de classe pour répercuter le nouvel emplacement, comme suit :

```
// Dans Formes/Carre.as :
class Shapes.Square {}

// Dans Formes/Cercle.as :
class Shapes.Circle {}

// Dans Formes/Triangle.as :
class Shapes.Triangle {}
```

Pour faire référence à une classe qui réside dans un répertoire de package, vous pouvez soit spécifier son nom complet, soit importer son package à l'aide de l'instruction `import`. Pour plus d'informations, voir « Fonctionnement des packages », à la page 202.

Comparaison des classes et des packages

En programmation orientée objet, une classe définit une catégorie d'objets. Les classes sont avant tout des types de données que vous pouvez créer pour définir un nouveau type d'objet dans votre application. La classe décrit les *propriétés* (données) et les *comportements* (méthodes) d'un objet, comme un plan d'architecte décrit les caractéristiques d'un immeuble.
Les propriétés (variables définies au sein d'une classe) et les méthodes d'une classe sont collectivement appelées *membres* de la classe. Pour utiliser les propriétés et méthodes définies par une classe, vous devez tout d'abord créer une occurrence de cette classe (sauf pour les classes comportant tous les membres statiques (voir « Présentation des membres (statiques) de classe », à la page 276, telle que la classe de niveau supérieur Math, et« Méthodes et propriétés statiques », à la page 223). La relation entre une occurrence et sa classe est similaire à la relation entre une maison et ses plans.

Dans Flash, les packages sont des répertoires qui contiennent un ou plusieurs fichiers de classe et résident dans un chemin de classe désigné. Placez tous les fichiers de classes personnalisées connexes dans un même répertoire Par exemple, vous pourriez avoir trois classes appelées AcierWidget, PlastiqueWidget et BoisWidget définies dans AcierWidget.as, PlastiqueWidget.as et BoisWidget.as et les organiser dans le package Widget. Pour plus d'informations sur les packages, consultez les sections « Fonctionnement des packages », à la page 202 et « Création et mise en package de vos fichiers de classe », à la page 241.

Fonctionnement des packages

Les packages sont des répertoires qui contiennent un ou plusieurs fichiers de classe et résident dans un répertoire de chemin de classe désigné. Par exemple, le package flash.filters est un répertoire sur votre disque dur contenant plusieurs fichiers de classe pour chaque type de filtre (tels que BevelFilter, BlurFilter, DropShadowFilter, etc.) dans Flash 8.

REMARQUE

Pour utiliser l'instruction `import`, vous devez spécifier ActionScript 2.0 et Flash Player 6 ou une version plus récente dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA.

L'instruction `import` permet d'accéder aux classes sans spécifier leur nom complet, avec qualificatifs. Par exemple, si vous souhaitez utiliser la classe BlurFilter dans un script, vous devez y faire référence avec son nom suivi de tous ses attributs (flash.filters.BlurFilter) ou l'importer. Si vous l'importez, vous pouvez y faire référence avec son nom de classe (BlurFilter). Le code ActionScript suivant démontre les différences entre l'utilisation de l'instruction `import` et l'utilisation des noms de classes complets.

Si vous n'importez pas la classe BlurFilter, votre code devra comporter les noms de classes complets (le nom du package suivi par le nom de classe) afin d'utiliser le filtre :

```
// sans importation
var myBlur:flash.filters.BlurFilter = new flash.filters.BlurFilter(10, 10,
  3);
```

Le même code, écrit avec une instruction `import` , vous permet d'accéder à BlurFilter en utilisant seulement le nom de classe au lieu de devoir toujours utiliser le nom complet du service. Cela peut éviter d'avoir à retaper, et réduire ainsi les éventuelles erreurs de typographies :

```
// avec importation
import flash.filters.BlurFilter;
var myBlur:BlurFilter = new BlurFilter(10, 10, 3);
```

Si vous importez plusieurs classes au sein d'un package (par exemple, BlurFilter, DropShadowFilter, et GlowFilter) il vous est possible d'utiliser l'une des deux méthodes pour importer chaque classe. La première méthode d'importation de classes multiples est d'importer chaque classe en utilisant une instruction `import` séparée, comme indiqué dans le fragment suivant :

```
import flash.filters.BlurFilter;
import flash.filters.DropShadowFilter;
import flash.filters.GlowFilter;
```

L'utilisation d'instructions `import` individuelles pour chaque classe dans un package peut prendre beaucoup de temps et conduire à des erreurs de typographie. La deuxième méthode d'importation de classes dans un package est d'utiliser un import générique, qui importe toutes les classes dans le niveau donné d'un package. L'ActionScript suivante montre un exemple d'utilisation d'un import générique :

```
import flash.filters.*; // importe chaque classe dans le package
  flash.filters
```

L'instruction `import` s'applique uniquement au script courant (image ou objet) dans lequel elle est appelée. Par exemple, supposons que vous deviez importer l'ensemble des classes du package macr.util dans l'image 1 d'un document Flash. Dans cette image, vous pouvez faire référence aux classes de ce package par leur nom de classe au lieu de leurs noms complets. Si vous souhaitez cependant utiliser le nom de classe dans un autre script d'image, vous devez faire référence aux classes de ce package par leur nom complet, ou ajouter une instruction `import` à l'autre image qui importe les classes dans ce package.

Lors de l'utilisation d'instructions `import`, Il est également important de noter que les classes sont importées seulement pour le niveau indiqué. Par exemple, si vous avez importé toutes les classes dans le package mx.transitions, seules les classes du répertoire /transitions/ sont importées, et non toutes les classes des sous-répertoires (telles les classes du package mx.transitions.easing).

CONSEIL	Si vous importez une classe, mais ne l'utilisez pas dans votre script, cette dernière n'est pas exportée avec le fichier SWF. Ceci signifie que vous pouvez importer des packages volumineux sans vous soucier de la taille du fichier SWF. Le pseudo-code binaire associé à une classe n'est inclus dans un fichier SWF que si cette classe est véritablement utilisée.

Présentation des valeurs et des types de données

Les données, les valeurs et les types sont des concepts importants lorsque vous commencez à écrire des classes et à les utiliser. Vous avez étudié les fichiers et types dans le Chapitre 3, « Données et types de données », à la page 35. Lorsque vous travaillez avec des classes, n'oubliez pas que les types de données décrivent la nature des informations qu'une variable ou qu'un élément ActionScript peut contenir, par exemple Boolean, Number et String. Pour plus d'informations, voir « Présentation des types de données », à la page 36.

Les expressions ont des valeurs alors que les valeurs et les propriétés ont des *types*. Les valeurs que vous pouvez définir et obtenir pour et à partir d'une propriété de votre classe doivent être compatibles avec cette propriété. La compatibilité de type signifie que le type d'une valeur est compatible avec le type utilisé, comme dans l'exemple suivant :

```
var myNum:Number = 10;
```

Pour plus d'informations sur le typage strict des données, consultez la section « Affectation des types de données et typage strict », à la page 45.

Bases de la programmation orientée objet.

Au cours des sections suivantes, vous allez vous familiariser avec la terminologie employée tout au long de ce chapitre avant de commencer à écrire du code ActionScript. Cette brève introduction aux principes du développement orienté objet vous aide à mieux comprendre les exemples et les sections de ce chapitre et du reste de ce manuel. Ces principes sont présentés de façon plus approfondie dans le reste de ce chapitre ; ainsi que les détails de leur implémentation dans Flash.

Les sections suivantes utilisent l'analogie d'un chat en démontrant comment les chats peuvent être comparés à des concepts de programmation orientée objet.

Objets

Pensez à un objet du monde réel, un chat, par exemple. Supposons qu'un chat ait des *propriétés* (ou états), telles que nom, âge et couleur, et des comportements, tels que dormir, manger et ronronner. Dans le monde de la programmation orientée objet, les objets ont également des propriétés et des comportements. En utilisant les techniques de programmation orientée objet, vous pouvez modeler un objet du monde réel (comme un chat) ou quelque chose de plus abstrait (un processus chimique, par exemple).

REMARQUE	Le mot *comportement* est utilisé ici au sens général et ne fait pas référence au panneau Comportements de l'environnement de programmation de Flash.

Pour plus d'informations sur les objets, consultez la section « Type de données Object », à la page 43.

Occurrences et membres de classe

Reprenons, si vous le voulez bien, l'analogie du chat. Supposons qu'il existe des chats de couleur, d'âge et de nom différents, qui mangent et ronronnent de façon différente. Indépendamment de leurs différences individuelles, tous les chats appartiennent à la même catégorie, qui en termes de programmation orientée objet serait appelée une classe : la classe des chats. Dans la terminologie de la programmation orientée objet, chaque chat est une *occurrence* de la classe Cat.

De même, en programmation orientée objet, une classe définit un modèle de type d'objets. Les caractéristiques et comportements qui appartiennent à une classe sont appelés *membres* de cette classe. Les caractéristiques (dans l'exemple du chat, nom, âge et couleur) sont appelées *propriétés* de la classe et sont représentées sous forme de variables. Les comportements (jouer, dormir) sont appelés *méthodes* de la classe et sont représentés sous forme de fonctions.

Pour plus d'informations sur les occurrences et membres de classe, consultez les sections « Présentation des membres de classe », à la page 224 et « Utilisation de membres de classe », à la page 228.

Héritage

L'un des principaux avantages de la programmation orientée objet est que vous pouvez créer des *sous-classes* de la classe ou *étendre* la classe. La sous-classe hérite alors de l'ensemble des propriétés et méthodes de la classe. La sous-classe définit en général des méthodes et propriétés supplémentaires ou annule les méthodes ou propriétés définies dans la super-classe. Les sous-classes peuvent également supplanter (apporter leurs propres définitions) les méthodes définies dans une super-classe.

L'un des principaux avantages de l'utilisation de la structure super-classe/sous-classe est qu'il est plus simple de réutiliser un code similaire entre divers classes. Par exemple, vous pouvez créer une super-classe appelée Animal, contenant les caractéristiques et les comportements communs à tous les animaux. Vous pouvez ensuite créer plusieurs sous-classes qui héritent de la super-classe Animal et ajoutent les caractéristiques et les comportements d'un type d'animal spécifique.

Vous pouvez créer une classe Chat qui hérite d'une autre classe. Par exemple, vous créez une classe Mammifère qui définit certaines propriétés et certains comportements communs à tous les mammifères. Vous pouvez alors créer une sous-classe Chat qui permet d'étendre la classe Mammifère. Une autre sous-classe, la classe Siamois par exemple, peut étendre la classe Chat (*sous-classe*), et ainsi de suite.

L'écriture de sous-classes vous permet de réutiliser le code. Au lieu de créer à nouveau le code commun aux deux classes, il vous suffit d'étendre la classe existante.

CONSEIL	Dans une application complexe, la définition de la structure hiérarchique des classes représente une partie essentielle du processus de conception. Assurez-vous de bien définir cette hiérarchie avant de démarrer la programmation.

Pour plus d'informations sur les héritages et les sous-classes, consultez le Chapitre 7, « Héritage », à la page 281.

Interfaces

En programmation orientée objet, les *Interfaces* peuvent être décrites comme des modèles de définitions de classe,et les classes qui les implémentent servent à implémenter ce modèle de méthodes. En utilisant l'analogie du chat, une interface est similaire à un modèle de chat : Le modèle vous informe des parties dont vous avez besoin, mais pas nécessairement de quelle façon ces parties sont assemblées, ou comment elles travaillent.

Vous pouvez utiliser les interfaces pour ajouter de la structure et une facilité de maintenance à vos applications. Étant donné qu'ActionScript 2.0 ne prend en charge que l'extension d'une seule superclasse, vous pouvez utiliser des interfaces comme forme d'héritage multiple limité.

Une interface peut également être considérée comme un « contrat de programmation » destiné à imposer des relations entre des classes non connexes. Par exemple, supposons que vous travailliez avec une équipe de programmeurs et que chacun de vous travaille sur une partie (classe) différente de la même application. Lors de la réalisation de l'application, vous vous mettez d'accord sur un ensemble de méthodes que les différentes classes utiliseront pour communiquer. Ainsi, vous créez une interface qui déclare ces méthodes, leurs paramètres et leurs types de retour. Toute classe qui implémente cette interface doit fournir des définitions pour ces méthodes; dans le cas contraire, une erreur du compilateur se produit.

Pour plus d'informations sur les héritages, consultez la section Chapitre 7, « Héritage », à la page 281. Pour plus d'informations sur les interfaces, consultez le Chapitre 8, « Interfaces », à la page 295.

Encapsulation

Dans une conception orientée objet intelligente, les objets sont considérés comme des « boîtes noires » qui contiennent ou *encapsulent* des fonctionnalités. Un programmeur doit pouvoir interagir avec un objet en ne connaissant que ses propriétés, ses méthodes et ses événements (son interface de programmation), sans connaître les détails de son implémentation. Cette approche permet aux programmeurs d'atteindre un niveau supérieur d'abstraction et met en place un cadre de structuration pour la création de systèmes complexes.

L'encapsulation permet à ActionScript 2.0 d'inclure, par exemple, le contrôle de l'accès des membres, de sorte que les détails de l'implémentation restent privés et invisibles pour tout codage en dehors d'un objet. Le code situé en dehors d'un objet doit interagir avec l'interface de programmation de l'objet plutôt qu'avec les détails d'implémentation (qui peuvent être dissimulés dans les méthodes et propriétés privées). Cette approche présente des avantages importants. Par exemple, elle permet au créateur de l'objet de changer son implémentation sans modifier le code externe à l'objet, tant que l'interface de programmation reste inchangée.

Pour plus d'informations sur l'encapsulation, consultez la section « Utilisation de l'encapsulation », à la page 236.

Polymorphisme

La programmation orientée objet permet d'exprimer les différences entre les classes individuelles par une technique appelée *polymorphisme* qui permet aux classes de supplanter les méthodes de leurs super-classes et de définir des implémentations spécialisées de ces méthodes. Dans Flash, les sous-classes peuvent définir des implémentations spécialisées de méthodes héritées de sa superclasse, mais ne peuvent accéder à l'implémentation de la superclasse, comme dans d'autres langages de programmation.

Par exemple, vous pouvez commencer par une classe appelée Mammifère qui comporte les méthodes `play()` et `sleep()`. Vous créez ensuite les sous-classes Chat, Singe et Chien pour étendre la classe Mammifère. Les sous-classes supplantent la méthode `play()` de la classe Mammal, de façon à représenter de façon plus réaliste ces types d'animaux. La classe Singe implémente la méthode `play()` pour se balancer aux arbres ; la classe Chat implémente la méthode `play()` pour courir après une pelote ; la classe Chien implémente la méthode `play()` pour rapporter une balle. Dans la mesure où la fonctionnalité `sleep()` reste similaire quel que soit l'animal, vous utilisez l'implémentation de super-classe.

Pour plus d'informations sur le polymorphisme, consultez le Chapitre 7, « Héritage », à la page 281 et la section « Utilisation du polymorphisme dans une application », à la page 290.

Ecriture de fichiers de classe personnalisée

L'exemple suivant présente les différentes parties d'un fichier de classe. Vous apprenez à écrire une classe et à la modifier pour étendre son utilisation avec Flash. Vous découvrez également les différentes parties d'une classe et comment les importer, de même que des informations sur la manipulation des fichiers de classe personnalisée dans Flash.

Nous allons commencer par une classe très simple. L'exemple suivant présente l'organisation d'une classe simple appelée UserClass.

Pour définir une classe, utilisez le mot-clé `class` dans un fichier de script externe (et non dans le script en cours de rédaction dans le panneau Actions). La structure de la classe concerne également les fichiers d'interface. Cette structure est illustrée ci-dessous, suite à quoi vous allez créer une classe.

- Le fichier de classe commence par des commentaires de documentation qui incluent une description générale du code, en plus des informations sur l'auteur et la version.
- Ajoutez vos instructions `d'importation` (le cas échéant).
- Ecrivez une instruction de package, une déclaration de classe ou d'interface, comme suit : `class UserClass {...}`
- Incluez tout commentaire d'implémentation de classe ou d'interface nécessaire. Dans ces commentaires, ajoutez des informations pertinentes pour l'ensemble de la classe ou de l'interface.
- Ajoutez toutes vos variables statiques. Ecrivez les variables de classe publiques en premier, suivies des variables de classe privées.
- Ajoutez des variables d'occurrence. Ecrivez les variables de membre publiques en premier, suivies des variables de membre privées.

- Ajoutez l'instruction de constructeur, par exemple celui de l'exemple suivant :
  `public function UserClass(username:String, password:String) {...}`
- Ecrivez vos méthodes. Regroupez-les par fonctionnalités et non par accessibilité ou domaine. En organisant les méthodes de cette manière, vous améliorez la lisibilité et la clarté de votre code.
- Ecrivez les méthodes de lecture/définition dans le fichier de classe.

L'exemple suivant examine une simple classe ActionScript nommée User.

Pour créer des fichiers de classe :

1. Choisissez Fichier > Nouveau et sélectionnez Fichier ActionScript, puis cliquez sur OK.
2. Choisissez Fichier > Enregistrer sous et nommez le fichier **User.as**.
3. Saisissez le code ActionScript suivant dans la fenêtre de script :

```
/**
  Classe User
  auteur : John Doe
  version : 0.8
  modifié le : 08/21/2005
  copyright : Adobe Systems Incorporated

  Ce code définit une classe User personnalisée qui vous permet de créer
  de nouveaux utilisateurs et de spécifier leurs informations de
  connexion.
*/

class User {
  // Variables d'occurrence privées
  private var __username:String;
  private var __password:String;

  // Instruction de constructeur
  public function User(p_username:String, p_password:String) {
    this.__username = p_username;
    this.__password = p_password;
  }

  public function get username():String {
    return this.__username;
  }
  public function set username(value:String):Void {
    this.__username = value;
  }

  public function get password():String {
    return this.__password;
  }
  public function set password(value:String):Void {
    this.__password = value;
  }
}
```

4. Enregistrez les modifications apportées au fichier de classe.

   Le code précédent commence par un *commentaire de documentation* standard, qui indique le nom de la classe, l'auteur, la version, la date de la dernière modification, les informations de copyright et une brève description de la fonction de la classe.

   L'instruction de constructeur de la classe User demande deux paramètres : `p_username` et `p_password`, copiés dans les variables d'occurrence privés `__username` et `__password` de la classe. Le reste du code de la classe définit les propriétés de lecture et de définition pour les variables d'occurrence privées. Si vous voulez créer une propriété en lecture seule, définissez une fonction de lecture, et non de définition. Par exemple, pour être certain qu'un nom d'utilisateur ne peut plus être modifié après sa définition, supprimez la fonction de définition `username` dans le fichier de la classe User.

5. Choisissez Fichier > Nouveau, puis Document Flash.

6. Choisissez Fichier > Enregistrer sous et nommez le fichier **user_test.fla**. Enregistrez le fichier dans le même répertoire que User.as.

7. Saisissez le code ActionScript suivant sur l'image 1 du scénario :

```
import User;
var user1:User = new User("un1", "pw1");
trace("Before:");
trace("\t username = " + user1.username); // un1
trace("\t password = " + user1.password); // pw1
user1.username = "1nu";
user1.password = "1wp";
trace("After:");
trace("\t username = " + user1.username); // 1nu
trace("\t password = " + user1.password); // 1wp
```

   La classe User créée précédemment étant très simple, le code ActionScript du document Flash est également très direct. La première ligne de code importe la classe User personnalisée dans votre document Flash. L'importation de cette classe User permet d'utiliser la classe comme un type de données personnalisé.

   Une seule occurrence de la classe User est définie et affectée à une variable nommée `user1`. Vous affectez une valeur à l'objet User `user1` et définissez un `nom d'utilisateur` `un1` et un `mot de passe` `pw1`. Les deux instructions `trace` suivantes affichent la valeur actuelle de `user1.username` et `user1.password` via les fonctions de lecture de la classe User, qui retournent toutes les deux des chaînes. Les deux lignes suivantes utilisent les fonctions de définition de la classe User pour définir de nouvelles valeurs pour les variables `username` et `password`. Enfin, vous envoyez les valeurs des variables `username` et `password` au panneau de sortie. Les instructions `trace` affichent les valeurs modifiées que vous avez définies via les fonctions de définition.

8. Enregistrez le fichier FLA, puis choisissez Contrôle > Tester l'animation pour tester les fichiers.

   Les résultats des instructions `trace` s'affichent dans le panneau de sortie. Dans les exemples suivants, vous utilisez ces fichiers dans une application.

Pour des exemples qui démontrent comment créer un menu dynamique avec des données XML et un fichier de classe personnalisée, consultez les exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Cet exemple appelle le constructeur ActionScript `XmlMenu()` et lui transmet deux paramètres : Le chemin d'accès au fichier menu XML et une référence au scénario actuel. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/XML_Menu afin d'accéder à ces exemples.

- XmlMenu.as
- xmlmenu.fla

Utilisation des classes personnalisées dans une application

Dans la section « Ecriture de fichiers de classe personnalisée », à la page 208, vous avez créé un fichier de classe personnalisée. Dans les sections suivantes, vous utilisez ce fichier de classe dans une application. Le flux de travail de création de classes implique au minimum les étapes suivantes :

1. Définition d'une classe dans un fichier de classe ActionScript externe. Pour plus d'informations sur la définition et l'écriture d'un fichier de classe, consultez la section « Ecriture de fichiers de classe personnalisée », à la page 208.
2. Enregistrement du fichier de classe dans un répertoire de chemin de classe spécifique (emplacement où Flash recherche les classes) ou dans le même répertoire que le fichier FLA de l'application. Pour plus d'informations sur la définition du chemin de classe, consultez la section« Définition et modification du chemin de classe », à la page 214. Pour une comparaison et plus d'informations sur l'importation de classes, consultez la section « Importation de fichiers de classe », à la page 212.
3. Création d'une occurrence de la classe dans un autre script, soit dans un document FLA, soit dans un fichier de script externe, ou création d'une sous-classe basée sur la classe d'origine. Pour plus d'informations sur la création d'une occurrence de classe, consultez la section « Création d'occurrences de classes dans un exemple », à la page 255.

Les sections suivantes contiennent des exemples de code qui vous permettent de vous familiariser avec la création de classes dans ActionScript 2.0. Si vous ne maîtrisez pas ActionScript 2.0, consultez le Chapitre 3, « Données et types de données », à la page 35 et le Chapitre 4, « Eléments fondamentaux du langage et de la syntaxe », à la page 81.

Pour plus d'informations sur l'utilisation des classes personnalisées, consultez les rubriques suivantes :

- « Importation de fichiers de classe », à la page 212
- « Utilisation d'un fichier de classe dans Flash », à la page 218
- « Utilisation des méthodes et des propriétés d'un fichier de classe », à la page 219
- « Présentation des membres de classe », à la page 224
- « Présentation des méthodes de lecture et définition », à la page 229
- « Résolution des références de classe par le compilateur », à la page 217
- « Présentation des classes dynamiques », à la page 233
- « Utilisation de l'encapsulation », à la page 236
- « Utilisation du mot-clé this dans les classes », à la page 237

Pour des exemples qui démontrent comment créer un menu dynamique avec des données XML et un fichier de classe personnalisée, consultez les exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Cet exemple appelle le constructeur ActionScript `XmlMenu()` et lui transmet deux paramètres : le chemin au fichier menu XML et une référence au scénario. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/XML_Menu afin d'accéder à ces exemples.

- XmlMenu.as
- xmlmenu.fla

Importation de fichiers de classe

Pour pouvoir utiliser une classe ou une interface que vous avez définie, Flash doit pouvoir localiser les fichiers ActionScript externes qui contiennent la définition de la classe ou de l'interface ; il peut ainsi importer le fichier. La liste des répertoires dans lesquels Flash recherche les définitions de classe, d'interface, de fonction et de variable est appelée *chemin de classe*. Flash a deux paramètres de chemin de classe - un chemin de classe global et un chemin de classe au niveau du document :

- **Le chemin de classe global** est un chemin de classe partagé par tous les documents Flash. Vous le définissez dans la boîte de dialogue Préférences (Edition > Préférences (Windows) ou Flash > Préférences (Macintosh), cliquez sur ActionScript dans la liste Catégorie, puis sur Paramètres ActionScript 2.0).

- **Le chemin de classe au niveau du document** est un chemin de classe défini de façon spécifique pour un seul document Flash. Il est défini dans la boîte de dialogue Paramètres de publication (Fichier > Paramètres de publication, sélectionnez l'onglet Flash, puis cliquez sur le bouton Paramètres).

Lorsque vous importez des fichiers de classe, les règles suivantes s'appliquent :

- Les instructions `d'importation` peuvent se trouver aux emplacements suivants :
  - N'importe où avant la définition de classe des fichiers de classe
  - N'importe où dans les scripts d'image ou d'objet
  - N'importe où dans les fichiers ActionScript que vous incluez dans une application (en utilisant l'instruction `#include`).
- La syntaxe suivante permet d'importer des définitions individuelles mises en package :
  ```
  import flash.display.BitmapData;
  ```
- La syntaxe de caractère générique permet d'importer des packages entiers :
  ```
  import flash.display.*;
  ```

Vous pouvez également inclure le code ActionScript dans un fichier Flash (FLA) à l'aide d'une instruction `include`. Les règles suivantes s'appliquent à l'instruction `include` :

- Les instructions `include` sont pour l'essentiel un copier-coller du contenu dans le fichier ActionScript inclus.
- Les instructions `include` utilisées dans les fichiers ActionScript sont liées au sous-répertoire qui contient le fichier.
- Dans un document FLA, une instruction `include` ne peut apporter que du code valide dans les fichiers FLA. Cette restriction est également vraie pour les autres emplacements comprenant des instructions `include`. Par exemple, si vous avez une instruction `include` dans une définition de classe, seules les définitions de propriétés et de méthodes peuvent être présentes dans le fichier ActionScript inclus :

```
// Foo.as
class Foo {
  #include "FooDef.as"
}

// FooDef.as:
var fooProp;
function fooMethod() {}
trace("Foo"); // Cette instruction est interdite dans une définition
              // de classe.
```

Pour plus d'informations sur l'instruction `include`, consultez la section relative à la directive #include dans le *Guide de référence du langage ActionScript 2.0* : Pour plus d'informations sur les chemins de classe, consultez la section « Définition et modification du chemin de classe », à la page 214.

Définition et modification du chemin de classe

Pour utiliser une classe ou une interface que vous avez définie, Flash doit localiser les fichiers ActionScript externes qui contiennent sa définition. La liste des répertoires dans lesquels Flash recherche les définitions de classe et d'interface est appelée *chemin de classe*.

Lorsque vous créez un fichier de classe ActionScript, vous devez enregistrer le fichier dans l'un des répertoires spécifiés dans le chemin de classe, ou dans l'un de ses sous-répertoires. (Vous pouvez modifier le chemin de classe pour inclure le chemin souhaité.) Sinon, Flash ne sera pas en mesure de résoudre (localiser) la classe ou l'interface spécifiée dans le script. Les sous-répertoires que vous créez dans un répertoire de chemin de classe sont appelés packages. Ils vous permettent d'organiser vos classes. (Pour plus d'informations sur les packages, consultez la section « Création et mise en package de vos fichiers de classe », à la page 241.)

Flash dispose de deux paramètres de chemin de classe : un *chemin de classe global* et un *chemin de classe au niveau du document*. Le chemin de classe global est celui que partagent tous vos documents Flash. Le chemin de classe au niveau du document est celui qui est défini spécialement pour un seul document Flash.

Le chemin de classe global s'applique aux fichiers ActionScript externes et aux fichiers FLA ; vous les définissez dans la boîte de dialogue (Windows: Edition > Préférences (Windows) ou Flash > Préférences (Macintosh), puis sélectionnez ActionScript dans la liste Catégories, puis cliquez sur les Paramètres ActionScript 2.0). Vous pouvez définir le chemin de classe au niveau du document dans la boîte de dialogue Paramètres de publication (Fichier > Paramètres de publication, et sélectionnez l'onglet Flash, puis cliquez sur le bouton Paramètres).

REMARQUE

Lorsque vous cliquez sur le bouton de vérification de la syntaxe situé au-dessus de la fenêtre de script lors de la modification d'un fichier ActionScript, le compilateur n'examine que le chemin de classe global. Les fichiers ActionScript ne sont pas associés aux fichiers FLA en mode édition et ne disposent pas de leur propre chemin de classe.

Utilisation d'un chemin de classe global

Le chemin de classe global est celui que partagent tous vos documents Flash.

Vous pouvez modifier le chemin de classe global à l'aide de la boîte de dialogue Préférences. Pour modifier le paramètre de chemin de classe au niveau du document, utilisez la boîte de dialogue Paramètres de publication pour le fichier FLA. Dans les deux cas, vous pouvez ajouter des chemins de répertoires absolus (par exemple `C:/mes_classes`) et des chemins de répertoires relatifs (par exemple, `../mes_classes` ou « `.` »). L'ordre des répertoires dans la boîte de dialogue correspond à l'ordre de recherche.

Par défaut, le chemin de classe global contient un chemin absolu et un chemin relatif. Le chemin absolu est indiqué par $(LocalData)/Classes dans la boîte de dialogue Préférences. L'emplacement du chemin absolu est indiqué ici :

- Windows : Disque dur\Documents and Settings*utilisateur*\Local Settings\Application Data\Adobe\Adobe Flash CS3*langue*\Configuration\Classes.
- Macintosh : Disque dur/Utilisateurs/*utilisateur*/Library/Application Support/Adobe/Adobe Flash CS3/*langue*/Configuration/Classes

REMARQUE

Ne supprimez pas le chemin de classe global absolu. Flash l'utilise pour accéder aux classes intégrées. Si vous supprimez ce chemin de classe par inadvertance, rétablissez-le en ajoutant $(LocalData)/Classes comme nouveau chemin de classe.

La section chemin relatif du chemin de classe global est identifiée par un point (.) et cette section pointe vers le répertoire de documents actuel. Notez que les chemins de classe relatifs peuvent pointer vers des répertoires différents, en fonction de l'emplacement du document compilé ou publié.

La procédure suivante permet d'ajouter un chemin de classe global ou de modifier un chemin de classe existant.

Pour modifier le chemin de classe global :

1. Sélectionnez Edition > Préférences (Windows) ou Flash > Préférences (Macintosh) pour ouvrir la boîte de dialogue Préférences.
2. Dans la colonne de gauche, cliquez sur ActionScript, puis sur le bouton Paramètres d'ActionScript 2.0.
3. Cliquez sur le bouton Rechercher le chemin et localisez le dossier que vous souhaitez ajouter.
4. Rechercher le chemin que vous souhaitez ajouter, puis cliquez sur OK.

Pour supprimer un répertoire du chemin de classe :

1. Sélectionnez le chemin dans la liste des chemins de classe.
2. Cliquez sur le bouton Supprimer du chemin.

REMARQUE

Ne supprimez pas le chemin de classe global absolu. Flash l'utilise pour accéder aux classes intégrées. Si vous supprimez ce chemin de classe par inadvertance, vous pouvez le rétablir en ajoutant $(LocalData)/Classes comme nouveau chemin de classe.

Pour plus d'informations sur l'importation des packages, consultez la section « Fonctionnement des packages », à la page 202.

Utilisation d'un chemin de classe au niveau du document

Le chemin de classe au niveau du document ne s'applique qu'aux fichiers FLA. Vous définissez le chemin de classe au niveau du document dans la boîte de dialogue Paramètres de publication d'un fichier FLA spécifique (Fichier > Paramètres de publication, puis cliquez sur l'onglet Flash et sur les Paramètres d'ActionScript 2.0). Le chemin de classe au niveau du document est vide par défaut. Lorsque vous créez et enregistrez un fichier FLA dans un répertoire, ce dernier devient un répertoire du chemin de classe désigné.

Lorsque vous créez des classes, il peut être nécessaire de les placer dans un répertoire que vous ajoutez ensuite à la liste de répertoires de chemin de classe globaux dans les situations suivantes :

- Si vous avez un ensemble de classes d'utilitaires que tous vos projets utilisent
- Si vous souhaitez vérifier la syntaxe de votre code (cliquez sur le bouton Vérifier la syntaxe) dans le fichier ActionScript externe

La création d'un répertoire permet d'empêcher la perte de classes personnalisées si vous devez désinstaller puis réinstaller Flash ou si le répertoire du chemin de classe global est supprimé ou remplacé.

Par exemple, vous pouvez créer le répertoire suivant pour vos classes personnalisées :

- Windows : Hard Disk\Documents et Paramètres*utilisateur*\classes personnalisées.
- Macintosh : Hard Disk\Utilisateurs*utilisateur*\classes personnalisées.

Vous devez ensuite ajouter ce chemin à la liste de chemins de classe globaux (consultez la section « Utilisation d'un chemin de classe global », à la page 214).

Lorsque Flash tente de résoudre les références à des classes dans un script FLA, il recherche en premier le chemin de classe de niveau document spécifié pour ce fichier FLA. Si la classe ne figure pas dans ce chemin de classe ou si cette classe est vide, Flash recherche le chemin de classe global. Si Flash ne trouve pas la classe dans le chemin de classe global, une erreur de compilation se produit.

Pour modifier le chemin de classe au niveau du document :

1. Choisissez Fichier > Paramètres de publication pour ouvrir la boîte de dialogue Paramètres de publication.
2. Cliquez sur l'onglet Flash.
3. Cliquez sur le bouton Paramètres en regard du menu déroulant Version d'ActionScript.
4. Vous pouvez saisir le chemin du fichier manuellement ou cliquer sur le bouton Rechercher le chemin pour localiser le répertoire à ajouter au chemin de classe.

REMARQUE	Pour modifier un répertoire de chemin de classe existant, sélectionnez le chemin dans la liste Chemin de classe, cliquez sur le bouton Rechercher le chemin, ouvrez le répertoire que vous souhaitez ajouter et cliquez sur OK.

REMARQUE	Pour supprimer un répertoire du chemin de classe, sélectionnez le chemin dans la liste Chemin de classe, puis cliquez sur le bouton Supprimer le chemin sélectionné (-).

Pour plus d'informations sur les packages, consultez la section « Présentation des packages », à la page 200.

Résolution des références de classe par le compilateur

Lorsque Flash tente de résoudre les références de classe dans un script FLA, il recherche en premier le chemin de classe de niveau document spécifié pour ce fichier FLA. Si la classe ne figure pas dans ce chemin de classe ou si cette classe est vide, Flash recherche le chemin de classe global. Si la classe n'est pas trouvée dans le chemin de classe global, une erreur de compilation se produit.

Lorsque vous cliquez sur le bouton Vérifier la Syntaxe pendant la modification d'un fichier ActionScript, le compilateur consulte uniquement le chemin de classe global ; les fichiers ActionScript ne sont pas associés à des FLA en mode Edition et ne disposent pas de leur propre chemin de classe.

Utilisation d'un fichier de classe dans Flash

Pour créer une occurrence de classe ActionScript, utilisez l'opérateur `new` pour appeler la fonction constructeur de la classe. La fonction constructeur porte toujours le même nom que la classe dont elle renvoie une occurrence, que vous affectez généralement à une variable. Par exemple, si vous utilisez la classe User de la section « Ecriture de fichiers de classe personnalisée », à la page 208, vous pouvez écrire le code suivant pour créer un nouvel objet User :

```
var firstUser:User = new User();
```

REMARQUE

Dans certains cas, il n'est pas nécessaire de créer une occurrence de classe pour employer ses propriétés et ses méthodes. Pour plus d'informations sur les membres (statiques) de classe, consultez les sections « Présentation des membres (statiques) de classe », à la page 276 et « Méthodes et propriétés statiques », à la page 223.

Utilisez l'opérateur point (`.`) pour accéder à la valeur d'une propriété dans une occurrence. Entrez le nom de l'occurrence à gauche du point et le nom de la propriété à droite. Par exemple, dans l'instruction suivante, `firstUser` représente l'occurrence et `username` la propriété :

```
firstUser.username
```

Dans un document Flash, vous pouvez également utiliser les classes intégrées ou de niveau supérieur qui composent le langage ActionScript. Le code suivant crée par exemple un nouvel objet Array, puis affiche sa propriété `length` :

```
var myArray:Array = new Array("apples", "oranges", "bananas");
trace(myArray.length); // 3
```

Pour plus d'informations sur l'utilisation des classes personnalisées, consultez l'« Exemple : Utilisation de fichiers de classe personnalisée dans Flash », à la page 253. Pour plus d'informations sur les fonctions constructeur, consultez la section « Ecriture de la fonction constructeur », à la page 244.

Utilisation des méthodes et des propriétés d'un fichier de classe

En programmation orientée objet, les membres (propriétés ou méthodes) d'une classe peuvent être des membres d'occurrence ou de classe. Les membres d'occurrence sont créés pour chaque occurrence de la classe ; ils sont définis sur le prototype de la classe lorsqu'ils sont initialisés dans la définition de classe. Par contraste, les membres de classe sont créés une fois par classe (ils sont également appelés membres statiques).

Les propriétés sont des attributs qui définissent un objet. Par exemple, `length` est une propriété qui s'applique à tous les tableaux et spécifie le nombre d'éléments qu'ils contiennent. Les méthodes sont des fonctions associées à une classe. Pour plus d'informations sur les fonctions et les mÃ©thodes, consultez le Chapitre 5, « Fonctions et méthodes », à la page 171.

L'exemple suivant décrit la création d'une méthode dans un fichier de classe :

```
class Sample {
  public function myMethod():Void {
    trace("myMethod");
  }
}
```

Vous pourriez ensuite invoquer cette méthode dans votre document. Pour invoquer une méthode d'occurrence ou accéder à une propriété d'occurrence, faites référence à une occurrence de la classe. Dans l'exemple suivant, `picture01`, une occurrence de la classe personnalisée Picture (disponible dans l'exercice suivant), appelle la méthode `showInfo()` :

```
var img1:Picture = new Picture("http://www.helpexamples.com/flash/images/
  image1.jpg");
// Invocation de la méthode showInfo().
img1.showInfo();
```

L'exemple suivant présente l'écriture d'une classe personnalisée Picture qui stocke les divers éléments d'informations relatifs à une photo.

Pour utiliser les classes Picture et PictureClass dans un fichier FLA :

1. Choisissez Fichier > Nouveau, puis sélectionnez Fichier ActionScript. Enregistrez le document sous le nom **Picture.as** puis cliquez sur OK.

   Vous écrivez votre classe Picture personnalisée dans ce document.

2. Saisissez le code ActionScript suivant dans la fenêtre de script :

```
/**
  Classe Picture
  auteur : John Doe
  version : 0.53
  modifié le : 6/24/2005
  copyright : Adobe Systems Incorporated

  La classe Picture est utilisée comme conteneur d'une image et de son
  URL.
*/

class Picture {
  private var __infoObj:Object;

  public function Picture(src:String) {
    this.__infoObj = new Object();
    this.__infoObj.src = src;
  }

  public function showInfo():Void {
    trace(this.toString());
  }
  private function toString():String {
    return "[Picture src=" + this.__infoObj.src + "]";
  }

  public function get src():String {
    return this.__infoObj.src;
  }
  public function set src(value:String):Void {
    this.__infoObj.src = value;
  }
}
```

3. Enregistrez le fichier ActionScript.

4. Choisissez Fichier > Nouveau, puis Document Flash pour créer un nouveau fichier FLA. Enregistrez-le sous le nom **picture_test.fla** dans le même répertoire que le fichier de classe Picture.

5. Saisissez le code ActionScript suivant sur l'image 1 du scénario :

```
var picture1:Picture = new Picture("http://www.helpexamples.com/flash/
  images/image1.jpg");
picture1.showInfo();
this.createEmptyMovieClip("img_mc", 9);
img_mc.loadMovie(picture1.src);
```

6. Enregistrez le document Flash.
7. Sélectionnez Contrôle > Tester l'animation pour tester le document.

   Le texte suivant apparait dans le panneau de sortie :

```
[Picture src=http://www.helpexamples.com/flash/images/image1.jpg]
```

Pour des exemples qui démontrent comment créer un menu dynamique avec des données XML et un fichier de classe personnalisée, consultez les exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Cet exemple appelle le constructeur ActionScript `XmlMenu()` et lui transmet deux paramètres : le chemin au fichier menu XML et une référence au scénario. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/XML_Menu afin d'accéder à ces exemples.

- XmlMenu.as
- xmlmenu.fla

Présentation des méthodes et propriétés (membres) publiques, privées et statiques

Lorsque vous écrivez des fichiers de classe ActionScript dans un fichier de script externe, vous pouvez créer quatre types de méthodes et propriétés. méthodes et propriétés publiques, méthodes et propriétés privées, méthodes et propriétés publiques statiques, et propriétés et méthodes privées et statiques. Ces méthodes et propriétés définissent l'accès de Flash aux variables et permettent de spécifier les parties du code autorisées à accéder à certaines méthodes et propriétés.

Lorsque vous développez des applications, petites ou grandes, à base de classes, il est particulièrement important de déterminer si une méthode ou une propriété doit être privée ou publique. Ces précautions vous permettent de sécuriser votre code autant que possible. Par exemple, si vous développez une classe User, vous souhaiterez peut-être ne pas autoriser les utilisateurs qui s'en servent à modifier l'identifiant des utilisateurs. En définissant la propriété de la classe (parfois appelée *membre d'occurrence*) comme `privée`, vous pouvez limiter l'accès à la propriété au code de la classe ou à des sous-classes de cette classe, et empêcher ainsi tout autre utilisateur de la modifier directement.

Méthodes et propriétés publiques

Le mot-clé `public` indique qu'une variable ou une fonction est accessible à tout appelant. Les variables et les fonctions étant publiques par défaut, le mot-clé `this` est surtout utilisé pour des raisons de style et de lisibilité, indiquant que la variable existe dans le domaine actuel. Vous pouvez par exemple employer le mot-clé `this` pour des raisons de cohérence dans un bloc de code qui contient également des variables privées ou statiques. Le mot-clé `this` peut être utilisé avec le mot-clé public ou privé.

La classe Sample suivante présente déjà une méthode publique nommée `myMethod()` :

```
class Sample {
  private var ID:Number;
  public function myMethod():Void {
    this.ID = 15;
    trace(this.ID); // 15
    trace("myMethod");
  }
}
```

Pour ajouter une propriété `publique`, utilisez le mot-clé « public » au lieu de « private », comme dans l'exemple de code suivant :

```
class Sample {
  private var ID:Number;
  public var email:String;
  public function myMethod():Void {
    trace("myMethod");
  }
}
```

La propriété `email` étant publique, vous pouvez la modifier dans la classe Sample, ou directement dans un fichier FLA.

Méthodes et propriétés privées

Le mot-clé `private` spécifie qu'une variable ou une fonction est accessible uniquement par la classe qui la déclare ou la définit, ou par les sous-classes de cette classe. Par défaut, une variable ou une fonction est publique et accessible par tout appelant. Utilisez le mot-clé `this` lorsque vous souhaitez restreindre l'accès à une variable ou une fonction, comme dans l'exemple suivant :

```
class Sample {
  private var ID:Number;
  public function myMethod():Void {
    this.ID = 15;
    trace(this.ID); // 15
    trace("myMethod");
  }
}
```

Pour ajouter une propriété privée à la classe précédente, il vous suffit d'utiliser le mot-clé `private` avant le mot-clé `var`.

Si vous tentez d'accéder à la propriété `ID` privée depuis l'extérieur de la classe Sample, vous obtenez une erreur de compilation et une erreur de référence dans le panneau de sortie. Le message indique que le membre est privé et n'est pas accessible.

Méthodes et propriétés statiques

Le mot-clé `static` spécifie qu'une variable ou une fonction n'est créée qu'une fois par classe et non pas dans chaque objet basé sur cette classe. Vous pouvez accéder à un membre de classe statique sans créer une occurrence de sa classe. Les propriétés et méthodes statiques peuvent être définies dans le domaine public ou privé.

Les membres statiques, également appelés *membres de la classe*, sont affectés à la classe et non à n'importe quelle occurrence de la classe. Pour invoquer une méthode de classe ou accéder à une propriété de classe, faites référence au nom de la classe plutôt qu'à l'une de ses occurrences, comme indiqué dans le code suivant :

```
trace(Math.PI / 8); // 0.392699081698724
```

Si vous saisissez cette seule ligne de code dans la fenêtre de script du panneau Actions, le résultat s'affiche dans le panneau de sortie.

Par exemple, dans la classe Sample précédente, vous auriez pu créer une variable statique pour garder une trace du nombre d'occurrences créées de cette classe, comme dans le code suivant :

```
class Sample {
  public static var count:Number = 0;
  private var ID:Number;
  public var email:String;
  public function Sample() {
    Sample.count++;
    trace("count updated: " + Sample.count);
  }
  public function myMethod():Void {
    trace("myMethod");
  }
}
```

Chaque fois que vous créez une nouvelle occurrence de la classe Sample, la méthode constructeur suit le nombre total de classes Sample d'occurrence définies jusque-là.

Certaines classes ActionScript de niveau supérieur disposent de membres de classe (ou membres statiques), comme vous l'avez vu précédemment dans cette section lorsque vous avez appelé la propriété `Math.PI`. Pour invoquer les membres de classe (propriétés et méthodes) ou y accéder, vous utilisez le nom de la classe, et non l'une de ses occurrences. Il ne faut donc pas créer une occurrence de la classe pour utiliser ces propriétés et méthodes.

Par exemple, la classe de niveau supérieur Math comprend uniquement des méthodes et des propriétés statiques. Pour appeler ses méthodes, au lieu de créer une occurrence de la classe Math, appelez tout simplement les méthodes de la classe Math. Le code suivant appelle la méthode `sqrt()` de la classe Math :

```
var squareRoot:Number = Math.sqrt(4);
trace(squareRoot); // 2
```

Le code suivant appelle la méthode `max()` de la classe Math afin de déterminer lequel de deux nombres est le plus grand :

```
var largerNumber:Number = Math.max(10, 20);
trace(largerNumber); // 20
```

Pour plus d'informations sur la création de membres de classe, consultez les sections « Présentation des membres de classe », à la page 224 et « Utilisation de membres de classe », à la page 228.

Pour des exemples qui démontrent comment créer un menu dynamique avec des données XML et un fichier de classe personnalisée, consultez les exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Cet exemple appelle le constructeur ActionScript `XmlMenu()` et lui transmet deux paramètres : le chemin au fichier menu XML et une référence au scénario. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/XML_Menu afin d'accéder à ces exemples.

- XmlMenu.as
- xmlmenu.fla

Présentation des membres de classe

La plupart des membres (méthodes et propriétés) abordés jusqu'à présent dans ce chapitre font partie d'un type appelé *membre d'occurrence*. Pour chaque membre d'occurrence, il existe une copie unique de ce membre dans chaque occurrence de la classe. Par exemple, la variable du membre `email` de la classe Sample a un membre d'occurrence, dans la mesure où chaque personne a une adresse de messagerie différente.

Vous disposez également d'un autre type de membre, appelé *membre de classe*. Il ne peut y avoir qu'une seule copie d'un membre de classe, et vous l'utilisez pour l'ensemble de la classe. Toute variable déclarée dans une classe, mais en dehors d'une fonction, est une propriété de la classe. Dans l'exemple suivant, la classe Person a deux propriétés, `age` et `username`, respectivement de type chaîne et nombre.

```
class Person {
  public var age:Number;
  public var username:String;
}
```

De la même façon, toute fonction déclarée dans une classe est considérée comme étant une méthode de cette classe. Dans l'exemple de la classe Person, vous pouvez créer une méthode unique appelée `getInfo()` :

```
class Person {
  public var age:Number;
  public var username:String;
  public function getInfo():String {
    // définition de la méthode getInfo()
  }
}
```

Dans le code précédent, la méthode `getInfo()` de la classe Person, ainsi que les propriétés `age` et `username`, sont toutes des membres d'occurrence publics. La propriété `age` ne constituerait pas un bon membre de classe, car chaque personne a un âge différent. Seules les propriétés et les méthodes qui sont partagées par l'ensemble des individus de la classe doivent être des membres de classe.

Supposons que chaque classe doit disposer d'une variable appelée `species`, afin de donner le nom latin des espèces représentées par la classe. Pour chaque objet Person, l'espèce est *Homo sapiens*. Il serait fastidieux de stocker une copie unique de la chaîne « `Homo sapiens` » pour chaque occurrence de la classe, ce qui implique que ce membre soit un membre de classe.

Les membres de classe sont déclarés avec le mot-clé `static`. Par exemple, vous pouvez déclarer le membre de classe species avec le code suivant :

```
class Person {
  public static var species:String = "Homo sapiens";
  // ...
}
```

Vous pouvez également déclarer les méthodes d'une classe comme étant statiques, comme illustré par l'exemple de code suivant :

```
public static function getSpecies():String {
  return Person.species;
}
```

Les méthodes statiques ne peuvent accéder qu'aux propriétés statiques, pas aux propriétés d'occurrence. Par exemple, le code suivant génère une erreur de compilation, car la méthode de classe `getAge()` fait référence à la variable d'occurrence `age` :

```
class Person {
  public var age:Number = 15;
  // ...
  public static function getAge():Number {
    return age; /* **Erreur** : Il est impossible d'accéder aux variables
  d'occurrence dans des fonctions statiques. */
  }
}
```

Pour résoudre ce problème, vous pouvez soit faire de la méthode une méthode d'occurrence, soit faire de la variable une variable de classe.

Pour plus d'informations sur les membres de classe (ou propriétés statiques), consultez la section « Méthodes et propriétés statiques », à la page 223.

Pour des exemples qui démontrent comment créer un menu dynamique avec des données XML et un fichier de classe personnalisée, consultez les exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Cet exemple appelle le constructeur ActionScript `XmlMenu()` et lui transmet deux paramètres : le chemin au fichier menu XML et une référence au scénario en cours. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/XML_Menu afin d'accéder à ces exemples.

- XmlMenu.as
- xmlmenu.fla

Utilisation du modèle de conception Singleton

L'un des modes d'utilisation de membres de classe le plus répandu est le modèle de conception Singleton. Un *modèle de conception* définit une approche formelle destinée à structurer votre code. En général, vous concevez un modèle de structure pour résoudre un problème de programmation courant. Il existe de nombreux modèles de conception, tels que Singleton. Ce modèle permet de s'assurer que chaque classe ne dispose que d'une seule occurrence et offre une méthode globale d'accès à l'occurrence. Pour des informations détaillées sur le modèle de conception Singleton, consultez le site www.adobe.com/devnet/coldfusion/articles/design_patterns.html.

Certaines situations nécessitent un objet d'un type spécifique dans un système. Par exemple, un jeu d'échec implique un échiquier unique et chaque pays n'a qu'une seule capitale. Bien qu'il n'y ait qu'un seul objet, il est préférable d'encapsuler sa fonctionnalité dans une classe. Il peut être néanmoins nécessaire de gérer une occurrence de cet objet et d'y accéder. L'utilisation d'une variable globale est une possibilité, mais les variables globales ne sont pas souhaitables dans la plupart des projets. La meilleure approche consiste à laisser la classe gérer l'occurrence unique de l'objet à l'aide de membres de classe. L'exemple suivant montre une utilisation typique de modèle de conception Singleton, où l'occurrence Singleton est créée une seule fois.

Pour utiliser le modèle de conception Singleton :

1. Choisissez Fichier > Nouveau, puis sélectionnez Fichier ActionScript. Enregistrez le document sous le nom **Singleton.as**.
2. Saisissez le code ActionScript suivant dans la fenêtre de script :

```
/**
  Classe Singleton
  auteur : John Doe
  version : 0.53
  modifié le : 6/24/2008
  copyright : Adobe Systems Incorporated
*/

class Singleton {
  private static var instance:Singleton = null;
  public function trackChanges():Void {
    trace("tracking changes.");
  }
  public static function getInstance():Singleton {
    if (Singleton.instance == null) {
      trace("creating new Singleton.");
      Singleton.instance = new Singleton();
    }
    return Singleton.instance;
  }
}
```

3. Enregistrez le document Singleton.as.
4. Choisissez Fichier > Nouveau, puis sélectionnez Document Flash pour créer un nouveau fichier FLA, et enregistrez-le sous le nom **singleton_test.fla** dans le même répertoire que celui du fichier de classe singleton.
5. Saisissez le code ActionScript suivant sur l'image 1 du scénario :

```
Singleton.getInstance().trackChanges(); // Suivi des modifications.

var s:Singleton = Singleton.getInstance(); // Suivi des modifications.
s.trackChanges();
```

6. Enregistrez le document Flash.
7. Sélectionnez Contrôle > Tester l'animation pour tester le document.

L'objet Singleton n'est pas créé avant d'être nécessaire, lorsqu'une autre section de code l'appelle par l'intermédiaire de la méthode `getInstance()`. Cette formule est généralement appelée *création en dilettante* ; elle peut accroître l'efficacité de votre code dans certains cas.

Souvenez-vous de ne pas utiliser trop ou trop peu de fichiers de classes dans votre application, dans la mesure où ceci peut déboucher sur de nombreux fichiers de classe mal conçus et nuire aux performances de l'application ou du flux de travail. Vous devriez toujours essayer d'utiliser des fichiers de classe au lieu de placer du code à d'autres endroits (par exemple des scénarios) ; cependant, évitez de créer beaucoup de classes avec peu de fonctionnalités, ou peu de classes gérant de nombreuses fonctionnalités. Ces deux scénarios peuvent indiquer une conception médiocre.

Utilisation de membres de classe

Vous pouvez notamment utiliser des membres de classe (statiques) pour conserver les informations d'état sur une classe et ses occurrences. Supposons par exemple que vous souhaitez consigner le nombre d'occurrences créées à partir d'une classe donnée. Vous pouvez facilement y parvenir en utilisant une propriété de classe incrémentée à chaque création d'une nouvelle occurrence.

Dans l'exemple suivant, vous créez une classe nommée Gadget qui définit un compteur d'occurrences statiques unique nommé `widgetCount`. A chaque création d'une nouvelle occurrence de la classe, la valeur de `widgetCount` est incrémentée de `1` et la valeur actuelle de `widgetCount` est affichée dans le panneau de sortie.

Pour créer un compteur d'occurrence en utilisant une variable de classe :

1. Choisissez Fichier > Nouveau et sélectionnez Fichier ActionScript, puis cliquez sur OK.
2. Saisissez le code suivant dans la fenêtre de script :

```
class Widget {
  // Initialisation de la variable de classe
  public static var widgetCount:Number = 0;
  public function Widget() {
    Widget.widgetCount++;
    trace("Creating widget #" + Widget.widgetCount);
  }
}
```

   La variable `widgetCount` est déclarée comme statique ; elle est donc initialisée à 0 une seule fois. Chaque fois que l'instruction constructeur de la classe Widget est appelée, elle ajoute 1 à la variable `widgetCount`, puis affiche le numéro de l'occurrence en cours de création.
3. Enregistrez votre fichier sous le nom **Widget.as**.
4. Choisissez Fichier > Nouveau, puis sélectionnez Document Flash pour créer un nouveau document FLA, et enregistrez-le sous le nom **widget_test.fla** dans le même répertoire que Widget.as.

5. Dans widget_test.fla, saisissez le code suivant sur l'image 1 du scénario :

```
//Avant de créer une occurrence de la classe,
// Widget.widgetCount affiche zéro (0).
trace("Widget count at start: " + Widget.widgetCount); // 0
var widget1:Widget = new Widget(); // 1
var widget2:Widget = new Widget(); // 2
var widget3:Widget = new Widget(); // 3
trace("Widget count at end: " + Widget.widgetCount); // 3
```

6. Enregistrez les modifications du fichier widget_test.fla.
7. Choisissez Contrôle > Tester l'animation pour tester le fichier.

   Flash donne les résultats suivants dans le panneau Sortie :

```
Widget count at start: 0
Creating widget # 1
Creating widget # 2
Creating widget # 3
Widget count at end: 3
```

Présentation des méthodes de lecture et définition

Les méthodes de lecture/définition sont des méthodes d'accesseur, c'est-à-dire qu'elles constituent généralement une interface publique qui permet de changer les membres de classe privés. Vous utilisez les méthodes de lecture/définition pour définir une propriété. Vous accédez à ces méthodes comme à des propriétés extérieures à la classe, même si vous les définissez au sein de la classe en tant que méthodes. Ces propriétés externes à la classe peuvent porter un nom différent de celui de la propriété dans la classe.

Le choix des méthodes de lecture/définition présente quelques avantages, notamment la possibilité de créer des membres avec une fonctionnalité complexe auxquels vous pouvez accéder comme à des propriétés. Elles permettent également de créer des propriétés en lecture et écriture seules.

Bien que ces méthodes soient très pratiques, il est préférable de ne pas en *abuser* car, entre autres problèmes, elles risquent de compliquer la maintenance du code. De même, elles offrent un accès à l'implémentation de votre classe, comme les membres publics. La programmation orientée objet décourage l'accès direct aux propriétés à l'intérieur d'une classe.

Lorsque vous écrivez des classes, vous êtes encouragés à rendre autant que possible toutes vos variables d'occurrence privées et à ajouter des méthodes de définition et de lecture en conséquence. En effet, il arrive souvent que vous ne souhaitiez pas autoriser les utilisateurs à modifier certaines variables à l'intérieur de vos classes. Par exemple, lorsqu'une méthode statique privée surveille le nombre d'occurrences créées pour une classe spécifique, vous ne souhaitez pas qu'un utilisateur puisse modifier ce compteur à l'aide du code. Seule l'instruction constructor doit pouvoir incrémenter cette variable à chaque appel. Dans ce cas, vous pouvez créer une variable d'occurrence privée et n'autoriser qu'une méthode de lecture pour la variable du compteur. Les utilisateurs peuvent ainsi récupérer la valeur actuelle juste à l'aide de la méthode de lecture, et non pas définir de nouvelles valeurs via la méthode de définition. Créer une méthode de lecture sans associer de méthode de définition est un moyen simple de mettre certaines des variables de votre classe en lecture seule.

Utilisation des méthodes de lecture et définition

La syntaxe des méthodes de lecture et de définition est la suivante :

- Une méthode de lecture ne réclame aucun paramètre et renvoie toujours une valeur.
- Une méthode de définition réclame toujours un paramètre et ne renvoie jamais de valeur.

Les classes définissent généralement des méthodes de lecture qui fournissent un accès en lecture et des méthodes de définition qui fournissent un accès en écriture à une propriété donnée. Imaginons par exemple une classe contenant une propriété nommée `userName` :

```
private var userName:String;
```

Au lieu de permettre aux occurrences de la classe d'accéder directement à cette propriété (`user.userName = "Buster"`, par exemple), la classe peut utiliser deux méthodes, `getUserName()` et `setUserName()`, qui seront implémentées dans l'exemple suivant.

Utilisation de méthodes de lecture et définition :

1. Choisissez Fichier > Nouveau et sélectionnez Fichier ActionScript, puis cliquez sur OK.
2. Saisissez le code suivant dans la fenêtre de script :

```
class Login {
  private var __username:String;
  public function Login(username:String) {
    this.__username = username;
  }
  public function getUserName():String {
    return this.__username;
  }
  public function setUserName(value:String):Void {
    this.__username = value;
  }
}
```

3. Enregistrez le document ActionScript sous le nom de **Login.as**.

   Comme vous pouvez le constater, `getUserName()` renvoie la valeur actuelle de `userName` et `setUserName()` définit le paramètre de chaîne transmis à la méthode en tant que valeur de `userName`.

4. Choisissez Fichier > Nouveau, puis sélectionnez Document Flash pour créer un nouveau document FLA, et enregistrez-le sous le nom **login2_test.fla** dans le même répertoire que Login.as.

5. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
var user:Login = new Login("RickyM");

// Appel de la méthode getUserName()
var userName:String = user.getUserName();
trace(userName); // RickyM

// Appel de la méthode setUserName()
user.setUserName("EnriqueI");
trace(user.getUserName()); // EnriqueI
```

6. Choisissez Contrôle > Tester l'animation pour tester le fichier.

   Flash donne les résultats suivants dans le panneau Sortie :

```
RickyM
EnriqueI
```

Si, toutefois, vous préférez une syntaxe plus concise, vous pouvez utiliser des méthodes de lecture/définition implicites. Celles-ci permettent d'accéder directement aux propriétés de classe, tout en conservant de bonnes pratiques de programmation orientée objet.

Pour définir ces méthodes, utilisez les attributs de méthodes `get` et `set`. Créez des méthodes qui obtiennent ou définissent la valeur d'une propriété et ajoutez le mot-clé `get` ou `set` avant le nom de méthode, comme dans l'exemple suivant.

REMARQUE

Les méthodes de lecture et définition implicites sont des abréviations syntaxiques de la méthode `Object.addProperty()` dans ActionScript 1.0.

Utilisation de méthodes de lecture et définition implicites :

1. Choisissez Fichier > Nouveau et sélectionnez Fichier ActionScript, puis cliquez sur OK.
2. Saisissez le code suivant dans la fenêtre de script :

```
class Login2 {
  private var __username:String;
  public function Login2(username:String) {
    this.__username = username;
  }
  public function get userName():String {
    return this.__username;
  }
  public function set userName(value:String):Void {
    this.__username = value;
  }
}
```

3. Enregistrez le document ActionScript **sous le nom Login2.as**.

   N'oubliez pas que les méthodes de lecture ne réclament aucun paramètre. Une méthode set (définition) doit prendre exactement un paramètre requis. Une méthode de définition peut avoir le même nom qu'une méthode de lecture dans le même domaine. Les méthodes de lecture et définition n'ont pas le même nom que les autres propriétés. Par exemple, dans le code ci-dessus qui définit des méthodes de lecture et de définition nommées `userName`, vous ne pourriez pas avoir de propriété nommée `userName` dans la même classe.
4. Choisissez Fichier > Nouveau, puis sélectionnez Document Flash pour créer un nouveau document FLA, et enregistrez-le sous le nom **login2_test.fla** dans le même répertoire que Login2.as.
5. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
var user:Login2 = new Login2("RickyM");

// appel de la méthode « get »
var userNameStr:String = user.userName;
trace(userNameStr); // RickyM

// appel de la méthode « get »
user.userName = "EnriqueI";
trace(user.userName); // EnriqueI
```

   Contrairement aux méthodes ordinaires, les méthodes de lecture et de définition sont appelées sans parenthèses ni arguments. Les méthodes de lecture et de définition sont appelées de la même manière qu'une propriété.

6. Enregistrez le document Flash, puis choisissez Contrôle > Tester l'animation pour tester le fichier.

   Flash donne les résultats suivants dans le panneau Sortie :

```
RickyM
EnriqueI
```

REMARQUE	Vous ne pouvez pas utiliser d'attributs de méthode de lecture/définition dans des déclarations de méthode d'interface.

Présentation des classes dynamiques

L'ajout du mot-clé `dynamic` à une définition de la classe spécifie que les objets basés sur la classe spécifiée peuvent ajouter des propriétés dynamiques et y accéder pendant la période d'exécution. Il est préférable de ne créer de classes dynamiques que lorsque cette fonctionnalité est réellement nécessaire.

La vérification du type des classes dynamiques est moins stricte que pour les classes non dynamiques, dans la mesure où les membres sollicités au sein de la définition de classe et dans les occurrences de classe ne sont pas comparées à celles qui sont définies dans le domaine de la classe. Les fonctions des membres de classe, cependant, peuvent toujours faire l'objet d'une vérification du type de renvoi ou de paramètre.

Pour plus d'informations sur la création de classes dynamiques, consultez la section « Création de classes dynamiques », à la page 233.

Création de classes dynamiques

Par défaut, les propriétés et méthodes d'une classe sont fixes. C'est-à-dire que l'occurrence d'une classe ne peut créer ou accéder à des propriétés ou méthodes qui n'étaient pas déclarées ou définies à l'origine par la classe. Considérons par exemple une classe Person qui définit deux propriétés, `userName` et `age`.

Pour créer une classe non dynamique :

1. Choisissez Fichier > Nouveau et sélectionnez Fichier ActionScript, puis cliquez sur OK.
2. Saisissez le code ActionScript suivant dans la fenêtre de script :

```
class Person {
  public var userName:String;
  public var age:Number;
}
```

   Si, dans un autre script, vous créez une occurrence de la classe Personne et essayez d'accéder à une propriété de la classe qui n'existe pas, le compilateur génère une erreur.
3. Enregistrez le fichier dans votre disque dur sous le nom **Person.as**.
4. Choisissez Fichier > Nouveau, puis Document Flash pour créer un nouveau fichier FLA, puis cliquez sur OK.
5. Choisissez Fichier > Enregistrez sous, nommez le fichier **person_test.fla**, et enregistrez-le dans le même répertoire que la classe Person créée précédemment.
6. Ajoutez le code suivant pour créer une nouvelle occurrence de la classe Person (`firstPerson`) et tentez d'affecter une valeur à une propriété nommée `hairColor` (qui n'existe pas dans la classe Person).

```
var firstPerson:Person = new Person();
firstPerson.hairColor = "blue"; // Erreur. Il n'existe aucune propriété
                                // nommée 'hairColor'.
```

7. Enregistrez le document Flash.
8. Choisissez Contrôle > Tester l'animation pour tester le code.

   Ce code crée une erreur de compilation car la classe Person ne déclare pas de propriété nommée `hairColor`. Dans la plupart des cas, c'est exactement ce que vous souhaitez qu'il se passe. Les erreurs de compilation peuvent sembler indésirables, mais se révèlent souvent bénéfiques pour les programmeurs. Les bons messages d'erreur vous aident à écrire votre code correctement en mettant les erreurs en évidence de manière précoce dans le processus de programmation.

Dans certains cas, cependant, il peut être utile d'ajouter et d'accéder à des propriétés ou méthodes d'une classe à l'exécution qui ne sont pas définies dans la définition de classe originale. Le modificateur de classe dynamique vous permet de le faire.

Pour créer une classe dynamique :

1. Choisissez Fichier > Nouveau et sélectionnez Fichier ActionScript, puis cliquez sur OK.
2. Choisissez Fichier > Enregistrer sous et nommez le fichier **Person2.fla**. Enregistrez le fichier sur votre disque dur.
3. Saisissez le code suivant dans la fenêtre de script :

```
dynamic class Person2 {
  public var userName:String;
  public var age:Number;
}
```

   Ce code ActionScript ajoute le mot-clé `dynamic` à la classe Person de l'exemple précédent. Les occurrences de la classe Person2 peuvent désormais ajouter et accéder aux propriétés et méthodes qui ne sont pas définies dans cette classe.
4. Enregistrez les modifications apportées au fichier ActionScript.
5. Choisissez Fichier > Nouveau, puis Document Flash pour créer un nouveau fichier FLA, puis cliquez sur OK.
6. Choisissez Fichier > Enregistrer sous et nommez le fichier **person2_test.fla**. Enregistrez le fichier dans le même répertoire que Person2.as.
7. Ajoutez le code suivant pour créer une nouvelle occurrence de la classe Person2 (`firstPerson`) et affecter une valeur à une propriété nommée `hairColor` (qui n'existe pas dans la classe Person2).

```
var firstPerson:Person2 = new Person2();
firstPerson.hairColor = "blue";
trace(firstPerson.hairColor); // blue
```

8. Enregistrez les modifications apportées au fichier person2_test.fla.
9. Choisissez Contrôle > Tester l'animation pour tester le code.

   La classe personnalisée Flash étant dynamique, vous pouvez lui ajouter des méthodes et des propriétés au moment de l'exécution (pendant la lecture du fichier SWF). Lorsque vous testez le code, le texte `blue` s'affiche dans le panneau de sortie.

Lorsque vous développez des applications, il est préférable de ne pas rendre les classes dynamiques si ce n'est pas vraiment nécessaire. L'une des raisons en est que cette vérification du type des classes dynamiques est moins stricte que celle des classes non dynamiques, dans la mesure où les membres sollicités au sein de la définition de classe et dans les occurrences de classe ne sont pas comparées à celles qui sont définies dans le domaine de la classe.
Les fonctions des membres de la classe, cependant, peuvent toujours faire l'objet d'une vérification du type de renvoi ou de paramètre.

Les sous-classes des classes dynamiques sont également dynamiques, à une exception près. Les sous-classes de la classe MovieClip ne sont pas dynamiques par défaut, bien que la classe MovieClip elle-même le soit. Cette implémentation permet de bénéficier de davantage de contrôle sur les sous-classes de la classe MovieClip, car vous pouvez rendre vos sous-classes dynamiques si vous le souhaitez.

```
class A extends MovieClip {} // A n'est pas une sous-classe dynamique
dynamic class B extends A {} // B est une sous-classe dynamique
class C extends B {} // C est une sous-classe dynamique
class D extends A {} // D n'est pas une sous-classe dynamique
dynamic class E extends MovieClip{} // E est une sous-classe dynamique
```

Pour plus d'informations sur les sous-classes, consultez le Chapitre 7, « Héritage », à la page 281.

Utilisation de l'encapsulation

Dans une conception orientée objet intelligente, les objets sont considérés comme des « boîtes noires » qui contiennent ou *encapsulent* des fonctionnalités. Un programmeur doit pouvoir interagir avec un objet en ne connaissant que ses propriétés, ses méthodes et ses événements (son interface de programmation), sans connaître les détails de son implémentation. Cette approche permet aux programmeurs d'atteindre un niveau supérieur d'abstraction et met en place un cadre de structuration pour la création de systèmes complexes.

L'encapsulation permet à ActionScript 2.0 d'inclure, par exemple, le contrôle de l'accès des membres, de sorte que les détails de l'implémentation restent privés et invisibles pour tout codage en dehors d'un objet. Le code situé en dehors d'un objet doit interagir avec l'interface de programmation de l'objet plutôt qu'avec les détails d'implémentation. Cette approche présente des avantages importants. Par exemple, elle permet au créateur de l'objet de changer son implémentation sans modifier le code externe à l'objet, tant que l'interface de programmation reste inchangée.

L'encapsulation dans Flash consiste par exemple à définir tous vos membres et toutes les variables de classe comme privés et à forcer les utilisateurs qui implémentent vos classes à accéder à ces variables via les méthodes de définition et de lecture. Cette méthode d'encapsulation permet, si vous avez un jour besoin de modifier la structure des variables, de modifier uniquement le comportement des fonctions de lecture et de définition plutôt que d'obliger chaque développeur à changer sa méthode d'accès aux variables de la classe.

Le code suivant décrit la modification de la classe Person utilisée dans les exemples précédents, définit ses membres d'occurrence comme privés et des méthodes de lecture et de définition pour les membres d'occurrence privés :

```
class Person {
  private var __userName:String;
  private var __age:Number;
  public function get userName():String {
    return this.__userName;
  }
  public function set userName(value:String):Void {
    this.__userName = value;
  }
  public function get age():Number {
    return this.__age;
  }
  public function set age(value:Number):Void {
    this.__age = value;
  }
}
```

Utilisation du mot-clé this dans les classes

Utilisez le mot-clé `this` en tant que préfixe au sein de vos classes pour les méthodes et les variables de membre. Bien que non indispensable, ce mot-clé `this` permet d'indiquer facilement si une propriété ou une méthode appartient à une classe lorsqu'elle comporte un préfixe. En l'absence de ce mot-clé, vous ne pouvez pas savoir si la propriété ou la méthode appartient à la super-classe.

Vous pouvez également utiliser un préfixe de nom de classe pour les variables statiques et les méthodes, y compris dans une classe. Ceci permet de qualifier vos références et de rendre le code plus lisible. Selon l'environnement de programmation utilisé, l'ajout de préfixes permet également de bénéficier de conseils de code.

REMARQUE

Ces préfixes sont facultatifs et certains développeurs les considèrent superflus. Il est recommandé d'ajouter le mot clé `this` en tant que préfixe, dans la mesure où il améliore la lisibilité et permet d'écrire un code explicite en fournissant le contexte des méthodes et des variables.

Exemple : Ecriture de classes personnalisées

Maintenant que vous connaissez les bases d'un fichier de classe et les types d'éléments qu'il contient, il est temps de connaître quelques directives générales liées à la création d'un tel fichier. Le premier exemple de ce chapitre décrit l'écriture des classes et leur mise en package. Le second exemple présente l'utilisation de ces fichiers de classe avec un fichier FLA.

ATTENTION	Le code ActionScript des fichiers externes est compilé dans un fichier SWF lors de la publication, de l'exportation, du test ou du débogage d'un fichier FLA. Cela signifie que si vous apportez des modifications à un fichier externe, vous devez enregistrer le fichier et recompiler tous les fichiers FLA qui l'utilisent.

Comme nous l'avons vu dans la section « Ecriture de fichiers de classe personnalisée », à la page 208, une classe comprend deux parties : la *déclaration* et le *corps*. La déclaration de la classe comporte au minimum l'instruction class, suivie de l'identifiant du nom de la classe, puis des accolades d'ouverture et de fermeture ({}). Tout ce qui figure entre les accolades constitue le corps de la classe, comme indiqué dans l'exemple suivant :

```
class className {
  // corps de la classe
}
```

N'oubliez pas que vous pouvez définir des classes uniquement dans des fichiers ActionScript externes. Par exemple, vous ne pouvez pas définir de classe dans un script d'image d'un fichier FLA. Vous allez donc créer un nouveau fichier pour cet exemple.

Dans sa forme la plus simple, une déclaration de classe comprend le mot-clé `class`, suivi du nom de la classe (Person, dans ce cas), puis des accolades d'ouverture et de fermeture ({}). Tout ce qui est entre les accolades est appelé corps de la classe et c'est ici que les propriétés et méthodes de la classe sont définies.

A la fin de cet exemple, l'ordre de base de vos fichiers de classe est le suivant :

- Commentaires de documentation
- Déclaration de classe
- Fonction constructeur
- Corps de la classe

Nous n'écrivons pas de sous-classes dans ce chapitre. Pour plus d'informations sur les héritages et les sous-classes, consultez le Chapitre 7, « Héritage », à la page 281.

Cet exemple aborde les sujets suivants :

Pour des exemples qui démontrent comment créer un menu dynamique avec des données XML et un fichier de classe personnalisée, consultez les exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Cet exemple appelle le constructeur ActionScript `XmlMenu()` et lui transmet deux paramètres : le chemin au fichier menu XML et une référence au scénario en cours. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/XML_Menu afin d'accéder à ces exemples.

- XmlMenu.as
- xmlmenu.fla

Directives générales sur la création d'une classe

Voici quelques conseils à suivre lors de l'écriture de fichiers de classe personnalisés. Ils vous aideront à écrire des classes correctes et bien constituées. Vous mettez en pratique ces directives dans les exemples suivants.

- De manière générale, ne placez qu'une seule déclaration par ligne. Ne placez pas le même type de déclaration ou des types différents sur une même ligne. Formatez vos déclarations comme dans l'exemple ci-dessous :

```
private var SKU:Number; // numéro de stock du produit (numéro
                        // d'identification)
private var quantity:Number; // Quantité du produit
```

- Initialisez les variables locales lorsque vous les déclarez, à moins que cette valeur initiale ne soit déterminée par un calcul. Pour plus d'informations sur l'initialisation des variables, consultez la section « Ajout de méthodes et de propriétés », à la page 245.

- Déclarez les variables avant de les utiliser (y compris dans les boucles). Par exemple, le code suivant pré déclare la variable d'itérateur de boucle (`i`) avant de l'utiliser dans la boucle `for` :

```
var my_array:Array = new Array("one", "two", "three");
var i:Number;
for (i = 0 ; i < my_array.length; i++) {
  trace(i + " = " + my_array[i]);
}
```

- Evitez d'utiliser des déclarations locales qui masquent des déclarations de niveau englobant. Par exemple, ne déclarez les variables qu'une seule fois, comme dans l'exemple ci-dessous :

```
// Code incorrect
var counter:Number = 0;
function myMethod() {
  var counter:Number;
  for (counter = 0; counter <= 4; counter++) {
    // instructions ;
  }
}
```

  Ce code déclare la même variable dans un bloc interne.

- N'affectez pas plusieurs variables à une valeur unique dans une instruction car la lecture devient alors difficile, comme dans l'exemple de code ActionScript suivant :

```
// Forme incorrecte
xPos = yPos = 15;
```

  ou

```
// Forme incorrecte
class User {
  private var m_username:String, m_password:String;
}
```

- Ne rendez pas publiques sans raison des variables d'occurrence ou des variables de membre ou de classe statique. Ces variables doivent être déclarées explicitement comme publiques avant d'être créées de cette façon.
- Définissez la plupart des variables de membre sur privé à moins qu'il n'y ait une bonne raison de les rendre publiques. Il est préférable, du point de vue de la conception, de définir les variables de membre sur privé et de restreindre leur accès par l'intermédiaire d'un petit groupe de fonctions de lecture et définition.

Appellation des fichiers de classe

Les noms de classes doivent être des identifiants : le premier caractère doit être une lettre, un soulignement (_) ou le signe dollar ($), et chaque caractère suivant doit être une lettre, un nombre, un soulignement ou le signe dollar. Il est recommandé de toujours tenter de limiter les noms de classe à des lettres.

Le nom de classe doit correspondre exactement au nom du fichier ActionScript qui le contient, majuscules comprises. Dans l'exemple suivant, si vous créez une classe nommée Rock, le fichier ActionScript qui contient sa définition doit s'appeler Rock.as :

```
// Dans le fichier Rock.as
class Rock {
  // Corps de la classe Rock
}
```

Dans la section suivante, vous créez et nommez une définition de classe. Pour créer, nommer et mettre en package les fichiers de classe, consultez la section « Création et mise en package de vos fichiers de classe », à la page 241. Pour plus d'informations sur l'appellation de fichiers de classe, consultez la section « Appellation des classes et des objets », à la page 724.

Création et mise en package de vos fichiers de classe

Dans cette section, vous allez créer, nommer et mettre en package vos fichiers de classe pour cet exemple (« Exemple : Ecriture de classes personnalisées », à la page 238). Les sections suivantes présentent l'écriture complète (et pourtant simple) de fichiers de classe. Pour des informations détaillées sur les packages, consultez les sections « Présentation des packages », à la page 200,« Comparaison des classes et des packages », à la page 202 et « Fonctionnement des packages », à la page 202.

Lorsque vous créez un fichier de classe, choisissez un emplacement de stockage pour le fichier. Au cours des étapes suivantes, vous allez enregistrer le fichier de classe et le fichier d'application FLA qui l'utilise dans le même répertoire pour plus de simplicité. Toutefois, si vous souhaitez vérifier la syntaxe, vous devez également indiquer à Flash comment localiser le fichier. De façon générale, lorsque vous créez une application, ajoutez le répertoire dans lequel vous stockez votre application et les fichiers de classe dans le chemin de classe Flash. Pour plus d'informations sur les chemins de classe, consultez la section « Définition et modification du chemin de classe », à la page 214.

Les fichiers de classe sont également appelés fichiers ActionScript (AS). Vous créez des fichiers AS dans l'outil de programmation Flash ou à l'aide d'un éditeur externe. Par exemple, Macromedia Dreamweaver peut créer des fichiers AS.

REMARQUE

Le nom d'une classe (ClassA) doit correspondre exactement au nom du fichier AS qui la contient (ClassA.as). Ce point est très important. Si ces deux noms ne correspondent pas exactement, y compris les majuscules, la compilation de la classe échoue.

Pour créer un fichier de classe et une déclaration de classe :

1. Choisissez Fichier > Nouveau, puis Document Flash pour créer un nouveau document FLA, puis cliquez sur OK.
2. Choisissez Fichier > Enregistrer sous et nommez le nouveau fichier **package_test.fla**, puis enregistrez le document Flash dans le répertoire actif.

   Ultérieurement, vous ajouterez du contenu dans ce document Flash.
3. Choisissez Fichier > Nouveau et sélectionnez Fichier ActionScript, puis cliquez sur OK.
4. Choisissez Fichier > Enregistrer sous, et créez un nouveau sous-répertoire nommé **com**, puis procédez comme suit :
   a. Dans le sous-dossier com, créez un nouveau sous-répertoire appelé **adobe**.
   b. Dans le sous-dossier adobe, créez un nouveau sous-répertoire appelé **utils**.
   c. Enregistrez le document ActionScript actif dans le répertoire utils et nommez le fichier **ClassA.as**.
5. Saisissez le code suivant dans la fenêtre de script :

   ```
   class com.adobe.utils.ClassA {
   }
   ```

   Le code précédant crée une nouvelle classe nommée ClassA dans le package com.adobe.utils.
6. Enregistrez le document ActionScript sous le nom ClassA.as.
7. Choisissez Fichier > Nouveau et sélectionnez Fichier ActionScript, puis cliquez sur OK.
8. Choisissez Fichier > Enregistrez sous, nommez le fichier **ClassB.as**, et enregistrez-le dans le même répertoire que le fichier ClassA.as créé précédemment.
9. Saisissez le code suivant dans la fenêtre de script :

   ```
   class com.adobe.utils.ClassB {
   }
   ```

   Le code précédant crée une nouvelle classe nommée ClassB dans le package com.adobe.utils.
10. Enregistrez les modifications apportées aux fichiers de classe ClassA.as et ClassB.as.

Les fichiers de classe que vous utilisez dans un fichier FLA sont importés dans un fichier SWF lors de sa compilation. Le code que vous écrivez dans un fichier de classe doit respecter une certaine méthodologie et un certain ordre, tous deux abordés dans les sections suivantes.

Si vous créez plusieurs classes personnalisées, utilisez des packages pour organiser vos fichiers de classe. Un package est un répertoire qui contient un ou plusieurs fichiers de classe et qui réside dans un répertoire de chemin de classe désigné. Les noms de classe doivent être pleinement qualifiés dans le fichier dans lequel ils sont déclarés : ils doivent indiquer le nom du répertoire (package) dans lequel ils sont enregistrés. Pour plus d'informations sur les chemins de classe, consultez la section « Définition et modification du chemin de classe », à la page 214.

Par exemple, une classe nommée `com.adobe.docs.YourClass` est stockée dans le répertoire `com/adobe/docs`. La déclaration de classe dans le fichier YourClass.as ressemble à cela :

```
class com.adobe.docs.YourClass {
  // Votre classe
}
```

REMARQUE

Vous écrirez la déclaration de classe qui reflète le répertoire de package dans la section suivante : « Exemple : Ecriture de classes personnalisées », à la page 238.

De ce fait, il est recommandé de planifier votre structure de packages avant de commencer la création de classes. En effet, si vous décidez de déplacer les fichiers de classe après leur création, vous devrez modifier les instructions de déclaration de classe pour indiquer leur nouvel emplacement.

Pour mettre vos fichiers de classe en package :

1. Choisissez le nom de package que vous désirez utiliser.

   Les noms de package doivent être intuitifs et facilement identifiables par les autres développeurs. N'oubliez pas que le nom du package correspond également à une structure de répertoires spécifique. Par exemple, les classes du package `com.adobe.utils` doivent être placées dans un dossier com/adobe/utils sur votre disque dur.

2. Créez la structure de répertoires requise après avoir choisi un nom de package.

   Par exemple, si votre package s'appelle `com.adobe.utils`, créez une structure de répertoires com/adobe/utils et placez vos classes dans le dossier utils.

3. Utilisez le préfixe com.adobe.utils pour toute classe créée dans ce package.

   Par exemple, si votre nom de classe était ClassA, son nom complet devrait être `com.adobe.utils.ClassA` au sein du fichier de classe `com/adobe/utils/ClassA.as`.

4. Si, par la suite, vous modifiez la structure de package, souvenez-vous de modifier non seulement la structure de répertoires, mais aussi le nom de package dans chaque fichier de classe, ainsi que dans chaque instruction d'importation ou référence à une classe au sein de ce package.

Pour poursuivre l'écriture des fichiers de classe, consultez la section « Ecriture de la fonction constructeur », à la page 244.

Ecriture de la fonction constructeur

Vous avez déjà découvert l'écriture de la déclaration de classe dans la section « Création et mise en package de vos fichiers de classe », à la page 241. Dans cette partie du chapitre, vous écrivez ce que l'on appelle la *fonction constructeur* du fichier de classe.

REMARQUE	Vous allez découvrir l'écriture des commentaires, des instructions et des déclarations dans les sections suivantes.

Les constructeurs sont des fonctions que vous utilisez pour initialiser (*définir*) les propriétés et les méthodes d'une classe. Par définition, les constructeurs sont des fonctions au sein d'une définition de classe qui portent le même nom que la classe. Par exemple, le code suivant définit une classe Person et implémente une fonction constructeur. En programmation orientée objet, la fonction constructeur initialise toutes les nouvelles occurrences d'une classe.

Un constructeur de classe est une fonction spéciale appelée automatiquement lorsque vous créez une occurrence de classe à l'aide de l'`opérateur new`. La fonction constructeur porte le même nom que la classe qui la contient. Par exemple, la classe Person créée précédemment contenait la fonction constructeur suivante :

```
// Fonction constructeur de la classe Person
public function Person (uname:String, age:Number) {
  this.__name = uname;
  this.__age = age;
}
```

Lorsque vous écrivez des fonctions constructeur, gardez à l'esprit les points suivants :

- Si aucune fonction constructeur n'est explicitement déclarée, c'est-à-dire, si vous ne créez pas de fonction dont le nom correspond à celui de la classe, le compilateur crée automatiquement une fonction constructeur vide.
- Une classe ne peut contenir qu'une seule fonction constructeur ; les fonctions constructeur étendues ne sont pas autorisées dans ActionScript 2.0.
- La fonction constructeur ne doit pas avoir de type de renvoi.

Le terme *constructeur* est généralement utilisé lorsque vous créez (instanciez) un objet en fonction d'une classe particulière. Les instructions suivantes sont des appels aux fonctions constructeur pour la classe de niveau supérieur Array et pour la classe Person personnalisée :

```
var day_array:Array = new Array("Sun", "Mon", "Tue", "Wed", "Thu", "Fri",
  "Sat");
var somePerson:Person = new Person("Tom", 30);
```

Vous allez ensuite ajouter une fonction spéciale appelée constructeur.

REMARQUE

L'exercice suivant est associé à l'« Exemple : Ecriture de classes personnalisées », à la page 238. Si vous ne souhaitez pas suivre l'exemple, il vous est possible de télàcharger les fichiers de classe depuis www.helpexamples.com/flash/learnas/classes/.

Pour ajouter les fonctions constructeur à vos fichiers de classe :

1. Ouvrez le fichier de classe ClassA.as dans l'outil de programmation Flash.
2. Modifiez les fichiers de classe existants de sorte qu'ils correspondent au code suivant (les modifications à effectuer apparaissent en gras) :

```
class com.adobe.utils.ClassA {
  function ClassA() {
    trace("ClassA constructor");
  }
}
```

   Le code précédent définit une méthode constructeur pour la classe ClassA.
   Ce constructeur envoie une chaîne simple au panneau de sortie qui vous avertit lorsqu'une nouvelle occurrence de la classe a été créée.
3. Ouvrez le fichier de classe ClassB.as dans l'outil de programmation Flash.
4. Modifiez le fichier de classe de sorte qu'il corresponde au code suivant (les modifications à effectuer apparaissent en gras) :

```
class com.adobe.utils.ClassB {
  function ClassB() {
    trace("ClassB constructor");
  }
}
```

5. Enregistrez les deux fichiers ActionScript avant de continuer.

Pour poursuivre l'écriture des fichiers de classe, consultez la section « Ajout de méthodes et de propriétés », à la page 245.

Ajout de méthodes et de propriétés

Pour créer des propriétés pour les classes ClassA et ClassB, utilisez le mot-clé `var` qui définit des variables.

REMARQUE

Les trois exercices suivants sont associés à l'« Exemple : Ecriture de classes personnalisées », à la page 238. Si vous ne souhaitez pas suivre l'exemple, il vous est possible de télécharger les fichiers de classe depuis www.helpexamples.com/flash/learnas/classes/.

Pour ajouter des propriétés aux classes ClassA et ClassB :

1. Ouvrez les fichiers ClassA.as et ClassB.as dans l'outil de programmation Flash.
2. Modifiez le fichier ClassA.as ActionScript, de sorte qu'il corresponde au code suivant (les modifications à effectuer apparaissent en gras) :

```
class com.adobe.utils.ClassA {
  static var _className:String;
  function ClassA() {
    trace("ClassA constructor");
  }
}
```

   Le bloc de code précédent ajoute une seule nouvelle variable statique, `_className`, qui contient le nom de la classe active.
3. Modifiez la classe ClassB et ajoutez la variable statique de sorte qu'elle corresponde au code précédent.
4. Enregistrez les deux fichiers ActionScript avant de continuer.

CONSEIL	Par convention, les propriétés de classe sont définies dans la partie supérieure du corps de la classe. Procédez ainsi simplifie la lecture du code, mais n'est pas obligatoire.

Utilisez la syntaxe à deux points (par exemple `var username:String` et `var age:Number`) dans la déclaration des variables. Ceci est un exemple de typage strict des données. Lorsque vous typez une variable à l'aide du format `var variableName:variableType`, le compilateur ActionScript s'assure que toute valeur affectée à cette variable correspond au type spécifié. Si le type de données correct n'est pas utilisé dans le fichier FLA qui importe cette classe, le compilateur renvoie une erreur. Pour plus d'informations sur le typage strict des données, consultez la section « Affectation des types de données et typage strict », à la page 45.

Les membres d'une classe se composent de propriétés (déclarations de variables) et de méthodes (déclarations de fonctions). Vous devez déclarer et définir toutes les propriétés et méthodes dans le corps de la classe (entre accolades [{}]), sinon une erreur se produit lors de la compilation. Pour plus d'informations sur les membres, consultez la section « Présentation des méthodes et propriétés (membres) publiques, privées et statiques », à la page 221.

Pour ajouter des méthodes aux classes ClassA et ClassB :

1. Ouvrez les fichiers ClassA.as et ClassB.as dans l'outil de programmation Flash.
2. Modifiez les fichiers de classe ClassA, de sorte qu'ils correspondent au code suivant (les modifications à effectuer apparaissent en gras) :

```
class com.adobe.utils.ClassA {
  static var _className:String;

  function ClassA() {
    trace("ClassA constructor");
  }
  function doSomething():Void {
    trace("ClassA - doSomething()");
  }
}
```

   Le bloc de code en gras crée une nouvelle méthode dans la classe, qui envoie une chaîne au panneau de sortie.

3. Dans ClassA.as, choisissez Outils > Vérifier la syntaxe pour vérifier la syntaxe de votre fichier ActionScript.

   Si des erreurs sont signalées dans le panneau de sortie, comparez le code de votre script au code final de l'étape précédente. Si vous ne parvenez pas à réparer l'erreur, copiez et collez le code complet dans la fenêtre de script avant de continuer.

4. Vérifiez la syntaxe de ClassB.as comme vous l'avez fait pour ClassA.as.

   Si des erreurs apparaissent dans le panneau de sortie, copiez et collez le code complet dans la fenêtre de script avant de continuer :

```
class com.adobe.utils.ClassB {
  static var _className:String;

  function ClassB() {
    trace("ClassB constructor");
  }
  function doSomething():Void {
    trace("ClassB - doSomething()");
  }
}
```

5. Enregistrez les deux fichiers ActionScript avant de continuer.

Vous pouvez initialiser des propriétés en ligne, c'est-à-dire lorsque vous les déclarez, avec des valeurs par défaut, comme dans l'exemple suivant :

```
class Person {
  var age:Number = 50;
  var username:String = "John Doe";
}
```

Lorsque vous initialisez des propriétés alignées, l'expression du côté droit de l'affectation doit être une constante de compilation. C'est à dire que l'expression ne peut pas faire référence à un élément paramétré ou défini au moment de l'exécution. Les constantes de compilation comprennent les chaînes littérales, les nombres, les valeurs booléennes, null et undefined, ainsi que les fonctions constructeur pour les classes de niveau supérieur suivantes : Array, Boolean, Number, Object et String.

Pour initialiser des propriétés alignées :

1. Ouvrez les fichiers ClassA.as et ClassB.as dans l'outil de programmation Flash.
2. Modifiez les fichiers de classe ClassA, de sorte qu'ils correspondent à l'ActionScript suivant (les modifications à effectuer apparaissent en gras) :

```
class com.adobe.utils.ClassA {
  static var _className:String = "ClassA";

  function ClassA() {
    trace("ClassA constructor");
  }
  function doSomething():Void {
    trace("ClassA - doSomething()");
  }
}
```

   La seule différence entre le fichier de classe existant et le bloc de code précédent est qu'une valeur est à présent définie pour la variable statique `_className`, « ClassA ».
3. Modifiez le fichier de classe ClassB et ajoutez la propriété alignée, en modifiant la valeur sur « ClassB ».
4. Enregistrez les deux fichiers ActionScript avant de continuer.

Cette règle s'applique uniquement aux variables d'occurrence (variables copiées dans chaque occurrence d'une classe). Elle ne s'applique pas aux variables de classe (variables qui appartiennent véritablement à la classe).

REMARQUE	Lorsque vous initialisez les tableaux alignés, un seul tableau est créé pour toutes les occurrences de la classe.

Pour poursuivre l'écriture de votre fichier de classe, consultez la section « Contrôle de l'accès des membres dans vos classes », à la page 249.

Contrôle de l'accès des membres dans vos classes

Par défaut, toute propriété ou méthode de classe est accessible à toute autre classe : tous les membres d'une classe sont considérés comme publics par défaut. Toutefois, dans certains cas, vous pouvez souhaiter que d'autres classes n'aient pas accès aux données ou aux méthodes d'une classe. Vous devez alors faire en sorte que ces membres deviennent privés, c'est-à-dire disponibles uniquement pour la classe qui les déclare ou les définit.

Pour spécifier des membres publics ou privés, utilisez l'attribut de membre public ou private. Par exemple, le code suivant déclare une variable privée (une propriété) et une méthode privée (une fonction). La classe suivante (LoginClass) définit une propriété privée nommée `userName` et une méthode privée nommée `getUserName()` :

```
class LoginClass {
  private var userName:String;
  private function getUserName():String {
    return this.userName;
  }
  // Constructeur :
  public function LoginClass(user:String) {
    this.userName = user;
  }
}
```

Les membres privés (propriétés et méthodes) sont uniquement accessibles à la classe qui définit ces membres et aux sous-classes de cette classe d'origine. Les occurrences de la classe d'origine ou celles des sous-classes de cette classe ne peuvent pas accéder aux propriétés et méthodes privées ; c'est-à-dire que les membres privés ne sont accessibles qu'au sein des définitions de classe, et non au niveau des occurrences. Dans l'exemple suivant, vous modifiez l'accès des membres à vos fichiers de classe.

REMARQUE

Cet exercice est associé à l'« Exemple : Ecriture de classes personnalisées », à la page 238. Si vous ne souhaitez pas suivre l'exemple, il vous est possible de télécharger les fichiers de classe depuis www.helpexamples.com/flash/learnas/classes/.

Pour contrôler l'accès des membres :

1. Ouvrez les fichiers ClassA.as et ClassB.as dans l'outil de programmation Flash.
2. Modifiez le fichier ClassA.as ActionScript, de sorte que son contenu corresponde au code ActionScript suivant (les modifications à effectuer apparaissent en gras) :

```
class com.adobe.utils.ClassA {
  private static var _className:String = "ClassA";

  public function ClassA() {
    trace("ClassA constructor");
  }
  public function doSomething():Void {
    trace("ClassA - doSomething()");
  }
}
```

   Le code précédent définit les deux méthodes (la constructeur ClassA et la méthode `doSomething()` comme publiques, ce qui signifie que les scripts externes peuvent y accéder. La variable statique `_className` est définie comme privée, ce qui signifie qu'elle n'est accessible qu'au sein de la classe et pas depuis des scripts externes.
3. Modifiez le fichier ActionScript ClassB.as et ajoutez les mêmes méthode et propriété que dans le fichier ClassA.
4. Enregistrez les deux fichiers ActionScript avant de continuer.

Une occurrence de ClassA ou ClassB ne peut pas accéder aux membres privés. Par exemple, le code suivant, ajouté à l'image 1 du scénario d'un fichier FLA, provoquerait une erreur de compilation indiquant que la méthode est privée et qu'il est impossible d'y accéder :

```
import com.adobe.utils.ClassA;
var a:ClassA = new ClassA();
trace(a._className); // Erreur. Le membre est privé et n'est pas accessible.
```

Le contrôle de l'accès des membres est une fonction de compilation uniquement ; à l'exécution, Flash Player ne fait aucune distinction entre les membres publics et privés.

Pour poursuivre l'écriture de votre fichier de classe, consultez la section « Documentation des classes », à la page 251.

Documentation des classes

L'insertion de commentaires dans vos classes et interfaces constitue pour les autres utilisateurs un élément essentiel de leur documentation. Par exemple, vous pouvez distribuer vos fichiers de classe dans la communauté Flash ou travailler avec une équipe de concepteurs ou de développeurs amenés à utiliser vos fichiers dans leur travail ou dans le cadre d'un projet en cours. La documentation permet aux autres utilisateurs de mieux comprendre l'objectif et les origines de la classe.

Un fichier d'interface ou de classe comporte généralement deux types de commentaire : les *commentaires de documentation* et les *commentaires d'implémentation*. Les commentaires de documentation décrivent le cahier des charges du code, mais pas son implémentation. Les commentaires d'implémentation décrivent le code ou l'implémentation de sections spécifiques du code. Ces deux types de commentaires utilisent des séparateurs légèrement différents. Les commentaires de documentation sont séparés par `/**` et `*/`, tandis que les commentaires d'implémentation sont séparés par `/*` et `*/`.

REMARQUE	Les commentaires de documentation ne constituent pas un élément de langage dans ActionScript 2.0. Cependant, ils constituent une manière commune de structurer des commentaires dans un fichier de classe que vous pouvez utiliser dans vos fichiers AS.

Servez-vous des commentaires de documentation pour décrire les interfaces, les classes, les méthodes et les constructeurs. Incluez un commentaire de documentation par classe, interface ou membre, et placez-le directement avant la déclaration.

Si vous vous devez ajouter des informations supplémentaires à vos commentaires de documentation, utilisez des commentaires d'implémentation (sous la forme de blocs de commentaires ou de commentaires sur une ligne, décrits dans la section « Présentation des commentaires », à la page 101). Les commentaires d'implémentation, si vous les ajoutez, suivent directement la déclaration.

REMARQUE	N'incluez pas les commentaires qui ne se rapportent pas directement à la classe en cours de lecture. Par exemple, n'incluez pas de commentaires décrivant le package correspondant.

REMARQUE	L'exercice suivant est associé à l'« Exemple : Ecriture de classes personnalisées », à la page 238. Si vous ne souhaitez pas suivre l'exemple, il vous est possible de télécharger les fichiers de classe depuis www.helpexamples.com/flash/learnas/classes/.

Pour documenter vos fichiers de classe :

1. Ouvrez les fichiers ClassA.as et ClassB.as dans l'outil de programmation Flash.
2. Modifiez le fichier de classe ClassA et ajoutez le nouveau code (les modifications à effectuer sont en gras) en haut du fichier de classe :

```
/**
  Classe ClassA
  version 1.1
  6/21/2005
  copyright Adobe Systems Incorporated
 */
class com.adobe.utils.ClassA {
  private static var _className:String = "ClassA";

  public function ClassA() {
    trace("ClassA constructor");
  }
  public function doSomething():Void {
    trace("ClassA - doSomething()");
  }
}
```

   Le code ci-dessus a ajouté un commentaire en haut du fichier de classe. Il est toujours conseillé d'ajouter des commentaires à votre code ActionScript et à vos fichiers Flash afin d'indiquer des informations très utiles telles que l'auteur de la classe, la date de la dernière modification, des informations de copyright ou tout problème/bogue éventuel pouvant être lié au fichier.
3. Ajoutez un commentaire similaire en haut du fichier ActionScript ClassB.as, en modifiant le nom de la classe et toute autre information nécessaire.
4. Enregistrez les deux fichiers ActionScript avant de continuer.

Vous pouvez également ajouter un bloc de commentaires ou des commentaires d'une seule ligne ou de fin de ligne dans le code de la classe. Pour plus d'informations sur la rédaction de commentaires pertinents, consultez la section « Rédaction de commentaires appropriés », à la page 728. Pour plus d'informations sur les commentaires, consultez « Commentaires sur une ligne », à la page 102, « Commentaires sur plusieurs lignes », à la page 102 et « Commentaires en fin de ligne », à la page 103.

Pour apprendre à utiliser ces fichiers de classe personnalisée dans un fichier SWF, consultez la section « Exemple : Utilisation de fichiers de classe personnalisée dans Flash », à la page 253.

Exemple : Utilisation de fichiers de classe personnalisée dans Flash

Cet exemple utilise des fichiers de classe répertoriés dans l'exemple : « Exemple : Ecriture de classes personnalisées », à la page 238 ; vous pouvez également les télécharger depuis www.helpexamples.com/flash/learnas/classes/. Si vous avez complété « Exemple : Ecriture de classes personnalisées », à la page 238, localisez ClassA.as et ClassB.as sur votre disque dur.

Le package du fichier de la classe ClassA portant le nom `com.adobe.utils.ClassA`, vous devez vous assurer d'avoir enregistré les fichiers de classe dans la structure de répertoires appropriée. Créez un sous-dossier appelé com dans le répertoire actif. Dans ce dossier com, créez un dossier nommé adobe. Enfin, ajoutez un troisième et dernier sous-répertoire nommé utils dans le dossier adobe. Enregistrez les deux fichiers de classe ClassA.as et ClassB.as dans ce dossier utils. Vous êtes maintenant prêt à commencer avec cet exemple.

Vous pouvez utiliser les classes personnalisées écrites dans le cadre de l'« Exemple : Ecriture de classes personnalisées », à la page 238 avec un fichier FLA. Dans cet exemple, vous utilisez les classes personnalisées pour créer une petite application dans Flash. Vos classes sont compilées dans le fichier SWF lorsque vous publiez le document, et l'ensemble fonctionne ensuite de concert. Dans les exercices suivants, vous allez découvrir les chemins de classe, l'utilisation des fichiers de classe dans votre application, ainsi que la manière d'importer des classes et des packages.

Pour poursuivre cet exemple, passez à la section « Importation de classes et de packages », à la page 253.

Importation de classes et de packages

Pour faire référence à une classe dans un autre script, vous devez faire précéder le nom de la classe par son nom de package. La combinaison du nom de la classe et de son chemin de package correspond au nom pleinement qualifié de la classe. Si une classe réside dans un répertoire de chemin de classe de premier niveau (et non dans un sous-répertoire du répertoire de chemin de classe), son nom de classe est également son nom pleinement qualifié.

Pour spécifier des chemins de package, utilisez une notation de type point (`.`) pour séparer les noms des répertoires de package. Les chemins de package sont hiérarchiques : chaque point représente un répertoire imbriqué. Supposons par exemple que vous créiez une classe nommée ClassName, qui réside dans un package `com/adobe/docs/learnAs2` dans votre chemin de classe. Pour créer une occurrence de cette classe, vous pouvez spécifier le nom complet de la classe.

Vous pouvez également utiliser le nom de classe complet pour taper vos variables, comme illustré dans l'exemple suivant :

```
var myInstance:com.adobe.docs.learnAs2.ClassName = new
  com.adobe.docs.learnAs2.ClassName();
```

Vous pouvez utiliser l'instruction `import` pour importer des paquets dans un script, ce qui vous permet d'utiliser le nom abrégé d'une classe à la place de son nom complet. Vous pouvez également utiliser le caractère générique `*` pour importer toutes les classes dans un package. Dans ce cas, vous n'avez pas besoin d'utiliser le nom de classe complet à chaque emploi de la classe.

Supposons par exemple que dans un autre script, vous ayez importé la classe ci-dessus avec l'instruction `import`, comme dans l'exemple suivant :

```
import com.adobe.docs.learnAs2.util.UserClass;
```

Par la suite, dans le même script, vous pouvez faire référence à cette classe par son nom abrégé, comme dans l'exemple suivant :

```
var myUser:UserClass = new UserClass();
```

Vous pouvez également utiliser le caractère générique (*) pour importer toutes les classes dans un package donné. Supposons que vous ayez un package nommé `com.adobe.utils` contenant deux fichiers de classe ActionScript, ClassA.as et ClassB.as. Dans un autre script, vous pouvez importer les deux classes dans ce package en utilisant le caractère générique, comme dans l'exemple suivant :

```
import com.adobe.utils.*;
```

L'exemple suivant indique que vous pouvez faire référence aux deux classes directement dans le même script :

```
var myA:ClassA = new ClassA();
var myB:ClassB = new ClassB();
```

L'instruction `import` s'applique uniquement au script actif (image ou objet) dans lequel elle est appelée. Si une classe importée n'est pas utilisée dans un script, elle n'est pas incluse dans le pseudo-code binaire du fichier SWF résultant, et n'est pas disponible dans les fichiers SWF susceptibles d'être chargés par le fichier FLA contenant l'instruction `import`.

REMARQUE

L'exercice suivant est associé à l'« Exemple : Utilisation de fichiers de classe personnalisée dans Flash », à la page 253 qui succède aux exemples « Exemple : Ecriture de classes personnalisées ». Si vous avez besoin de ClassA et ClassB, il vous est possible de télécharger les fichiers de classe depuis www.helpexamples.com/flash/learnas/classes/.

Pour importer une classe ou un package :

1. Ouvrez le fichier package_test.fla.
2. Saisissez le code suivant dans la fenêtre de script :

```
import com.adobe.utils.*;
var a = new ClassA(); // Constructeur ClassA
var b = new ClassB(); // Constructeur ClassB
```

   Le bloc de code précédent commence par importer chacune des classes dans le package `com.adobe.utils` via le caractère générique *. Ensuite, vous créez une nouvelle occurrence de la classe ClassA, ce qui entraîne l'envoi d'un message dans le panneau de sortie. Une occurrence de la classe ClassB est également créée, ce qui envoie également des messages de débogage au panneau de sortie.

3. Enregistrez les modifications apportées au document Flash avant de continuer.

Pour continuer à utiliser ces fichiers de classe dans un fichier Flash, consultez la section « Création d'occurrences de classes dans un exemple », à la page 255.

Création d'occurrences de classes dans un exemple

Les occurrences sont des objets qui contiennent toutes les propriétés et les méthodes d'une classe donnée. Par exemple, les tableaux étant des occurrences de la classe Array, vous pouvez utiliser n'importe quelle méthode ou propriété de la classe Array avec n'importe quelle occurrence de tableau. Vous pouvez également créer votre propre classe, par exemple UserSettings, puis créer une occurrence de la classe UserSettings.

Reprenons l'exemple commencé dans le cadre de l'« Exemple : Utilisation de fichiers de classe personnalisée dans Flash », à la page 253, lorsque vous avez modifié un fichier FLA pour importer les classes de manière à ne plus avoir à utiliser leurs noms complets.

Dans cet exemple (« Exemple : Utilisation de fichiers de classe personnalisée dans Flash », à la page 253), l'étape suivante consiste à créer une occurrence des classes ClassA et ClassB dans un script, telle qu'un script d'image dans un document Flash package_test.fla et de l'affecter à une variable. Pour créer une occurrence d'une classe personnalisée, utilisez l'opérateur `new`, comme pour créer une occurrence de classe ActionScript de niveau supérieur (telle que Date ou Array). Vous faites référence à la classe par son nom de classe complet, ou vous l'importez, comme dans la section « Importation de classes et de packages », à la page 253.

REMARQUE

L'exercice suivant est associé à l'« Exemple : Utilisation de fichiers de classe personnalisée dans Flash », à la page 253 qui succède aux exemples « Exemple : Ecriture de classes personnalisées ».

Pour créer une nouvelle occurrence des classes ClassA et ClassB :

1. Ouvrez le fichier **package_test.fla**.
2. Saisissez le code en gras suivant dans la fenêtre de script :

```
import com.adobe.utils.*;
var a:ClassA = new ClassA(); // Constructeur ClassA
a.doSomething(); // appel de la méthode doSomething() de ClassA
var b:ClassB = new ClassB(); // Constructeur ClassB
b.doSomething(); // appel de la méthode doSomething() de ClassB
```

   Le fait de saisir vos objets dans cet exemple de code permet au compilateur de s'assurer que vous n'essayez pas d'accéder aux propriétés ou aux méthodes qui ne sont pas définies dans votre classe personnalisée. Pour plus d'informations sur le typage strict des données, consultez la section « Affectation des types de données et typage strict », à la page 45. Une exception au typage strict de vos objets, serait de déclarer que la classe est dynamique à l'aide du mot clé `dynamic`. Voir « Création de classes dynamiques », à la page 233.
3. Enregistrez les modifications apportées au fichier FLA avant de continuer.

A présent, les bases de la création et de l'utilisation des classes dans les documents Flash ne devraient plus avoir de secrets pour vous. N'oubliez pas que vous pouvez également créer des occurrences des classes ActionScript de niveau supérieur ou intégrées (voir « Utilisation des classes intégrées », à la page 274).

Pour continuer à utiliser ces fichiers de classe dans un fichier Flash, consultez la section « Affectation d'une classe à des symboles dans Flash », à la page 256.

Affectation d'une classe à des symboles dans Flash

Vous pouvez également affecter une classe aux symboles susceptibles d'être utilisés dans un fichier Flash, tels qu'un clip sur la scène.

Pour affecter une classe à un symbole de clip :

1. Choisissez Fichier > Nouveau et sélectionnez Fichier ActionScript, puis cliquez sur OK.
2. Choisissez Fichier > Enregistrer sous, et nommez le fichier **Animal.as**, puis sauvegardez le fichier sur votre disque dur.
3. Saisissez le code suivant dans la fenêtre de script :

```
class Animal {
  public function Animal() {
    trace("Animal::constructor");
  }
}
```

   Ce code ActionScript crée une nouvelle classe appelée Animal qui dispose d'une méthode constructeur envoyant une chaîne au panneau de sortie.

4. Enregistrez les modifications apportées au fichier ActionScript.
5. Choisissez Fichier > Nouveau, puis Document Flash pour créer un nouveau fichier FLA, puis cliquez sur OK.
6. Choisissez Fichier > Enregistrez sous et nommez le fichier **animal_test.fla** ; enregistrez le fichier dans le même dossier que le fichier Animal.as créé à l'étape 2.
7. Choisissez Insertion > Nouveau Symbole pour ouvrir la boîte de dialogue Créer un symbole.
8. Saisissez le nom de symbole d'un **animal** et sélectionnez l'option Movie Clip.
9. Cliquez sur le bouton Avancé dans le coin inférieur de la boîte de dialogue Créer un nouveau symbole, pour activer davantage d'options.

   Le bouton Avancé est disponible lorsque vous êtes en mode basique de la boîte de dialogue Créer un nouveau symbole.
10. 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.
11. Saisissez une valeur pour l'identifiant **animal_id** et définissez la Classe ActionScript 2.0 sur **Animal** (selon le nom de classe spécifié à l'étape 3).
12. Sélectionnez la case à cocher Exporter dans la première image, puis cliquez sur OK pour appliquer vos modifications et fermer la boîte de dialogue.
13. Enregistrez le document Flash et choisissez Contrôle > Tester l'animation.

    Le texte provenant de la fonction constructeur de votre classe Animal apparaît dans le panneau de sortie.

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

Compilation et exportation de classes

Par défaut, les classes utilisées par un fichier SWF sont mises en packages et exportées dans la première image du fichier SWF. Vous pouvez également spécifier une image différente dans laquelle vos classes sont mises en package et exportées. Cette option est très pratique, par exemple lorsqu'un fichier SWF utilise de nombreuses classes dont le téléchargement est long (telles que des composants). Si les classes sont exportées vers la première image, l'utilisateur doit attendre que tout le code de classe soit téléchargé avant de voir apparaître cette image. En spécifiant une image ultérieure dans le scénario, vous pouvez afficher une courte animation de chargement dans les premières images du scénario, pendant le téléchargement du code de classe dans l'image ultérieure.

Pour spécifier l'image à exporter pour les classes d'un document Flash :

1. Choisissez Fichier > Nouveau, puis Document Flash. Enregistrez le nouveau document sous le nom **exportClasses.fla**.
2. Renommez le calque par défaut **content**, faites glisser un composant ProgressBar du panneau Composants vers la scène et donnez-lui **my_pb** comme nom d'occurrence.
3. Créez un nouveau calque, faites-le glisser sur le calque Contenu, et renommez-le **actions**.
4. Ajoutez le code ActionScript suivant à l'image 1 du calque actions du scénario principal :

```
my_pb.indeterminate = true;
```

5. Créez une nouvelle image-clé sur Image 2 du calque actions et ajoutez le code ActionScript suivant :

```
var classesFrame:Number = 10;
if (_framesloaded < classesFrame) {
  trace(this.getBytesLoaded() + " of " + this.getBytesTotal() + " bytes
  loaded");
  gotoAndPlay(1);
} else {
  gotoAndStop(classesFrame);
}
```

6. Créez une nouvelle image-clé sur Image 10 du calque actions et ajoutez le code ActionScript suivant :

```
stop();
```

7. Créez une nouvelle image-clé sur Image 10 du calque actions et faites glisser plusieurs composants sur la scène.
8. Cliquez avec le bouton droit sur chaque composant (masquez ProgressBar) dans le panneau Bibliothèque et choisissez Liaison dans le menu contextuel pour lancer la boîte de dialogue Propriétés de liaison.

9. Dans la boîte de dialogue Propriétés de liaison, assurez-vous que l'option Exporter pour ActionScript est sélectionnée, désélectionnez la boîte de dialogue Exporter dans la première image, et cliquez sur OK.
10. Choisissez Fichier > Paramètres de publication.
11. Dans la boîte de dialogue Paramètres de publication, cliquez sur l'onglet Flash.
12. Cliquez sur le bouton Paramètres qui se trouve en regard du menu contextuel de la version ActionScript pour ouvrir la boîte de dialogue Paramètres d'ActionScript.
13. Dans le champ de texte Exporter l'image pour les classes, saisissez le numéro de l'image cible (Image 10).

    Si l'image spécifiée n'existe pas dans le scénario, un message d'erreur apparaît lors de la publication du fichier SWF.
14. Cliquez sur OK pour fermer la boîte de dialogue Paramètres d'ActionScript, puis cliquez de nouveau sur OK pour fermer la boîte de dialogue Paramètres de publication.
15. Choisissez Contrôle > Tester l'animation pour tester le document Flash. Si les composants se chargent trop rapidement, sélectionnez View > Simuler le téléchargement depuis le fichier SWF. Flash simule le téléchargement du document Flash à une vitesse inférieure, qui vous permet de voir les composants de la barre de progression s'animer tandis que les fichiers de classe se téléchargent.

Pour plus d'informations sur les fichiers ASO, reportez-vous à « Utilisation des fichiers ASO », à la page 259.

Utilisation des fichiers ASO

Lors de la compilation, Flash crée parfois des fichiers portant l'extension .aso dans le sous-répertoire /aso du répertoire de chemin de classe global par défaut (consultez la section « Définition et modification du chemin de classe », à la page 214). L'extension .aso signifie *objet ActionScript* (ASO). Pour chaque fichier ActionScript 2.0 importé de façon implicite ou explicite et pour lequel la compilation est réussie, Flash génère un fichier ASO. Ce fichier contient le pseudo-code binaire issu du fichier ActionScript associé. Ces fichiers contiennent donc la forme compilée (le *pseudo-code binaire*) d'un fichier de classe.

Flash ne doit recréer le fichier ASO que dans les cas suivants :

- Le fichier AS correspondant a été modifié.
- Les fichiers ActionScript qui contiennent des définitions importées ou utilisées par le fichier ActionScript correspondant ont été modifiés.
- Les fichiers ActionScript inclus par le fichier ActionScript correspondant ont été modifiés.

Le compilateur crée des fichiers ASO pour faciliter la gestion du cache. La première compilation est plus lente que les compilations suivantes. Ceci est dû au fait que les fichiers AS ont changé et sont recompilés en tant que fichiers ASO. Pour les fichiers AS ne comportant pas de modifications, le compilateur lit le pseudo-code compilé directement dans le fichier ASO, sans recompiler le fichier AS.

Le format de fichier ASO est un format intermédiaire développé pour l'usage interne. Il n'est pas documenté et n'a pas été conçu pour la redistribution.

Si Flash semble compiler des versions plus anciennes d'un fichier que vous avez édité, supprimez les fichiers ASO et compilez-les de nouveau. Si vous prévoyez de supprimer des fichiers ASO, supprimez-les lorsque Flash n'effectue aucune autre opération, telle que la vérification de la syntaxe ou l'exportation de SWF.

Pour supprimer des fichiers ASO :

Si vous modifiez un fichier FLA et que vous souhaitez supprimer un fichier ASO, sélectionnez l'une des actions suivantes dans l'environnement auteur :

- Sélectionnez Contrôle > Supprimer fichiers ASO pour supprimer des fichiers ASO et poursuivre la modification.
- Sélectionnez Contrôle > Supprimer fichiers ASO et Tester l'animation pour supprimer des fichiers ASO et tester l'application.

Si vous modifiez un document ActionScript dans la fenêtre de script :

- Sélectionnez Contrôle > Supprimer fichiers ASO pour supprimer des fichiers ASO et poursuivre la modification.
- Sélectionnez Contrôle > Supprimer fichiers ASO et Tester le projet pour supprimer des fichiers ASO et tester l'application.

La quantité de code pouvant être insérée dans un classe n'est pas illimitée : Le pseudo-code d'une définition de classe dans un fichier SWF exporté ne doit faire plus de 32767octets. Un message d'avertissement s'affiche s'il dépasse cette limite.

Vous ne pouvez pas prédire la taille du pseudo-code d'une classe donnée, mais les classes comportant jusqu'à 1 500 lignes dépassent rarement la limite.

Si votre classe dépasse la limite, déplacez une partie du code dans une autre classe. En général, une programmation orientée objet de qualité consiste à créer des classes aussi concises que possible.

Distinction entre classes et domaine

Lorsque vous placez du code ActionScript dans des classes, il peut être nécessaire de modifier le mode d'utilisation du mot-clé `this`. Par exemple, si une méthode de classe utilise une fonction de rappel (telle que la méthode `onLoad()` de la classe LoadVars) dans votre code, il sera difficile de savoir si le mot-clé `this` fait référence à la classe ou à l'objet. Dans ce cas, il peut être nécessaire de créer un pointeur vers la classe actuelle, comme dans l'exemple suivant.

Pour distinguer le domaine et les fichiers de classe externes :

1. Choisissez Fichier > Nouveau et sélectionnez Fichier ActionScript, puis cliquez sur OK.
2. Saisissez ou collez le code suivant dans la fenêtre de script :

```
/**
  Classe Product
  Product.as
*/
class Product {
  private var productsXml:XML;
  // Constructeur
  // targetXmlStr - chaîne, contient le chemin d'un fichier XML
  function Product(targetXmlStr:String) {
    /* Crée une référence locale à la classe actuelle.
       Même si vous êtes au niveau du gestionnaire d'événement onLoad du
  code XML,
       vous pouvez faire référence à la classe actuelle et pas seulement
  au paquet XML.
    */
    var thisObj:Product = this;
    // Crée une variable locale qui permet de charger le fichier XML.
    var prodXml:XML = new XML();
    prodXml.ignoreWhite = true;
    prodXml.onLoad = function(success:Boolean) {
      if (success) {
        /* Lorsque le code XML se charge et est analysé correctement,
           définir la variable productsXml de la classe sur le document
           XML analysé et appeler la fonction init.
        */
        thisObj.productsXml = this;
        thisObj.init();
      } else {
        /* Une erreur s'est produite pendant le chargement du fichier
         XML. */
        trace("error loading XML");
      }
    };
    // Amorcer le chargement du document XML.
    prodXml.load(targetXmlStr);
  }
  public function init():Void {
    // Afficher le package XML.
    trace(this.productsXml);
  }
}
```

Dans la mesure où vous tentez de référencer la variable du membre privé au sein d'un gestionnaire `onLoad`, le mot-clé `this` fait en réalité référence à l'occurrence `prodXml` et non à la classe Product, à laquelle on s'attendrait. Par conséquent, vous devez créer un pointeur vers le fichier de classe local de façon à pouvoir référencer directement la classe à partir du gestionnaire `onLoad`. Vous pouvez maintenant utiliser cette classe dans un document Flash.

3. Enregistrez le code ActionScript précédent sous le nom **Product.as**.
4. Créez un nouveau document Flash nommé **testProduct.fla** dans le même répertoire.
5. Sélectionnez l'image 1 du scénario principal.
6. Saisissez le code ActionScript suivant dans le panneau Actions :

```
var myProduct:Product = new Product("http://www.helpexamples.com/
  crossdomain.xml");
```

7. Choisissez Contrôle > Tester l'animation pour tester ce code dans l'environnement de test.

   Le contenu du document XML spécifié s'affiche dans le panneau de sortie.

Les variables et les fonctions statiques sont un autre type de domaine rencontré avec l'utilisation de ces classes. Le mot-clé `static` indique qu'une variable ou une fonction n'est créée qu'une fois par classe et non pas dans chaque occurrence de cette classe. Vous pouvez accéder à un membre de classe statique sans créer d'occurrence de la classe en utilisant la syntaxe `someClassName.username`. Pour plus d'informations sur les variables et les fonctions statiques, consultez les sections « Présentation des méthodes et propriétés (membres) publiques, privées et statiques », à la page 221 et « Utilisation de membres de classe », à la page 228.

Un autre avantage des variables statiques est qu'elles ne perdent pas leurs valeurs à la fin de leur domaine. L'exemple suivant présente l'utilisation du mot-clé `static` pour créer un compteur chargé de suivre le nombre d'occurrences de la classe Flash créées. La variable `numInstances` étant statique, elle n'est créée qu'une fois pour l'ensemble de la classe, et pas pour chaque occurrence.

Pour utiliser le mot-clé static :

1. Choisissez Fichier > Nouveau et sélectionnez Fichier ActionScript, puis cliquez sur OK.
2. Saisissez le code suivant dans la fenêtre de script :

```
class User {
  private static var numInstances:Number = 0;
  public function User() {
    User.numInstances++;
  }
  public static function get instances():Number {
    return User.numInstances;
  }
}
```

   Le code précédent définit une classe User qui surveille le nombre d'appels au constructeur. Une variable statique privée (`User.numInstances`) est incrémentée dans la méthode constructeur.

3. Enregistrez le document sous le nom **User.as**.
4. Choisissez Fichier > Nouveau puis sélectionnez Document Flash pour créer un nouveau document FLA, et enregistrez-le dans le même répertoire que User.as.
5. Saisissez le code ActionScript suivant sur l'image 1 du scénario :

```
trace(User.instances); // 0
var user1:User = new User();
trace(User.instances); // 1
var user2:User = new User();
trace(User.instances); // 2
```

   La première ligne de code appelle la méthode de lecture statique `instances()` qui renvoie la valeur de la variable statique privée `numInstances`. Le reste du code crée de nouvelles occurrences de la classe User et affiche la valeur actuelle renvoyée par la méthode de lecture `instances()`.
6. Choisissez Contrôle > Tester l'animation pour tester les documents.

Pour plus d'informations sur l'utilisation du mot-clé `this` dans les classes, consultez la section « Utilisation du mot-clé this dans les classes », à la page 237.

Présentation des classes de niveau supérieur et intégrées

Outre les principaux éléments de langage, constructions (boucles `for` et `while`, par exemple) et types de données primitives (nombres, chaînes et valeurs booléennes) décrits précédemment dans ce manuel (voir le Chapitre 3, « Données et types de données », à la page 35 et le Chapitre 4, « Eléments fondamentaux du langage et de la syntaxe », à la page 81), ActionScript fournit également plusieurs classes intégrées (*types de données complexes*). Ces classes offrent de nombreuses options et fonctionnalités à vos scripts. Vous avez utilisé les classes de niveau supérieur et d'autres classes intégrées du langage ActionScript dans les chapitres précédents et vous allez encore les exploiter dans les chapitres restants.
De nombreuses classes fournies avec Flash permettent d'ajouter interactivité et fonctionnalité dans vos fichiers SWF ou encore de développer des applications complexes. Par exemple, vous pouvez utiliser la classe Math pour calculer des équations dans vos applications. Vous pouvez également utiliser la classe BitmapData pour créer des animations codées et des pixels.

Les classes de niveau supérieur, énumérées dans « Classes de niveau supérieur », à la page 265, sont écrites dans Flash Player. Dans la boîte à outils Actions, ces classes sont situées dans le répertoire ActionScript 2.0 Classes. Certaines d'entre elles reposent sur les spécifications ECMAScript (ECMA-262) edition 3 language specification et sont appelées *classes ActionScript de base*. Elles comprennent les classes Array, Boolean, Date et Math. Pour plus d'informations sur les packages, consultez la section « Fonctionnement des packages », à la page 202.

Vous pouvez trouver les classes ActionScript installées sur votre disque dur. Vous pouvez trouver les dossiers classes dans :

- Windows : Disque dur\Documents and Settings*utilisateur*\Local Settings\Application Data\Adobe\Flash CS3*langue*\Configuration\Classes.
- Macintosh : Disque dur/Utilisateurs/*utilisateur*/Library/Application Support/Adobe/Adobe Flash CS3/*langue*/Configuration/Classes

Le document Do note the Read Me placé dans ce répertoire pour plus d'information sur sa structure.

La distinction entre les classes ActionScript de base et les classes spécifiques à Flash est similaire à celle qui existe entre les classes JavaScript de base et côté client. Tout comme les classes JavaScript côté client permettent de contrôler l'environnement du client (le navigateur Web et le contenu des pages Web), les classes spécifiques à Flash permettent de contrôler l'apparence et le comportement d'une application Flash à l'exécution.

Les autres classes intégrées ActionScript sont spécifiques à Flash et au modèle d'objet Flash Player. Parmi ces classes, citons Camera, MovieClip et LoadVars. D'autres classes sont organisées en packages, telles que flash.display. Toutes ces classes sont parfois dites *intégrées* (classes prédéfinies pouvant être utilisées pour ajouter des fonctionnalités à vos applications).

Les sections suivantes présentent les classes ActionScript intégrées, et décrivent les tâches fondamentales que vous effectuez avec ces classes intégrées. Pour un aperçu de l'utilisation des classes et des objets en programmation orientée objet, consultez la section « Utilisation des classes intégrées », à la page 274. Les exemples de code en utilisant ces classes sont inclus dans tout le manuel.

Pour plus d'informations sur les éléments du langage (constantes, opérateurs et instructions), consultez le Chapitre 4, « Eléments fondamentaux du langage et de la syntaxe », à la page 81

Pour plus d'informations sur les classes intégrées et de niveau supérieur, consultez les rubriques suivantes :

Autres éléments du langage

Il y a d'autres éléments de langage qui constituent ActionScript, en dehors des classes. Ils incluent des directives, des constantes, des fonctions globales, des propriétés globales, des opérateurs et des instructions. Pour plus d'informations sur l'utilisation de chacun de ces éléments de langage, consultez les rubriques suivantes :

- Chapitre 4, « Eléments fondamentaux du langage et de la syntaxe »
- Chapitre 5, « Fonctions et méthodes »

Vous trouverez une liste de ces éléments de language dans les sections suivantes de *Guide de référence du langage ActionScript 2.0* :

- Directives de compilation
- Constantes
- Fonctions globales
- propriétés globales
- opérateurs
- Instructions

Classes de niveau supérieur

Le niveau supérieur contient les fonctions globales et les classes ActionScript qui, pour la plupart, fournissent les fonctionnalités de base de vos applications. *Les classes de base*, directement empruntées à ECMAScript, comprennent Array, Boolean, Date, Error, Function, Math, Number, Object, String et System. Pour plus d'informations sur chaque classe, consultez le tableau suivant.

REMARQUE

Les classes CustomActions et XMLUI sont uniquement disponibles dans l'environnement de programmation Flash.

Classe	Description
Accessibility	La classe Accessibility gère la communication entre les fichiers SWF et les applications de lecture d'écran. Conjointement avec la propriété globale `_accProps`, les méthodes de cette classe permettent de contrôler les propriétés accessibles des clips, des boutons et des champs de texte lors de l'exécution. Reportez-vous à Accessibilité.

Classe	Description
Array	La classe Array représente des tableaux dans ActionScript et tous les objets de tableau sont des occurrences de cette classe. Elle contient les méthodes et propriétés réservées aux objets de tableau. Reportez-vous à Array.
AsBroadcaster	Cette classe offre des capacités de notification d'événement et de gestion des écouteurs pouvant être ajoutées à d'autres objets. Reportez-vous à AsBroadcaster.
Boolean	La classe Boolean est une enveloppe pour les valeurs booléennes (`true` ou `false`). Reportez-vous à Boolean.
Button	La classe Button fournit des méthodes, propriétés et gestionnaires d'événement pour l'utilisation des boutons. Reportez-vous à Button. Notez que la classe intégrée Button est différente de la classe du composant Button, associée au composant Button version 2.
Camera	La classe Camera vous permet d'accéder à la caméra de l'utilisateur, s'il en a installé une. Utilisé avec Flash Media Server, votre fichier SWF peut saisir, diffuser et enregistrer les images et les vidéos de la caméra d'un utilisateur. Reportez-vous à Camera.
Color	La classe Color permet de définir la valeur des couleurs RVB et leur évolution dans les occurrences de clips, puis de récupérer ces valeurs après leur définition. La classe Color est abandonnée dans Flash Player 8 en faveur de la classe ColorTransform. Pour plus d'informations sur ColorTransform, se reporter à ColorTransform (flash.geom.ColorTransform).
ContextMenu	La classe ContextMenu conditionne le contenu du menu contextuel de Flash Player à l'exécution. Vous pouvez associer des objets ContextMenu distincts à des objets MovieClip, Button ou TextField à l'aide de la propriété `menu` de ces classes. Vous avez aussi tout loisir d'utiliser la classe ContextMenuItem pour ajouter des éléments de menu personnalisés à un objet ContextMenu. Reportez-vous à ContextMenu.
ContextMenuItem	La classe ContextMenuItem vous permet de créer des éléments de menu dans le menu contextuel de Flash Player. Pour ajouter les éléments de menu que vous créez à l'aide de cette classe au menu contextuel de Flash Player, vous utilisez la classe ContextMenu. Reportez-vous à ContextMenuItem.
CustomActions	La classe CustomActions vous permet de gérer toute action personnalisée enregistrée avec l'outil de programmation Reportez-vous à CustomActions.

Classe	Description
Date	La classe Date détermine la représentation des dates et des heures dans ActionScript et prend en charge leurs opérations de manipulation. La classe Date permet également de récupérer la date et l'heure actuelle du système d'exploitation. Reportez-vous à Date.
Error	La classe Error contient des informations sur les erreurs d'exécution survenant dans vos scripts. En règle générale, vous utilisez l'instruction `throw` pour générer une condition d'erreur, que vous pouvez ensuite gérer à l'aide d'une instruction `try.catch.finally`. Reportez-vous à Error.
Function	La classe Function est la représentation sous forme de classe de toutes les fonctions ActionScript, y compris celles qui sont natives d'ActionScript et celles que vous définissez. Reportez-vous à Function.
Key	La classe Key fournit des méthodes et propriétés permettant d'obtenir des informations sur les actions effectuées sur le clavier et les touches par l'utilisateur. Reportez-vous à Key.
LoadVars	La classe LoadVars permet de transférer des variables en paires nom-valeur d'un fichier SWF à un serveur. Reportez-vous à LoadVars.
LocalConnection	La classe LocalConnection permet de développer des fichiers SWF qui peuvent échanger des instructions entre eux sans utiliser la méthode `fscommand()` ni JavaScript. Reportez-vous à LocalConnection.
Math	La classe Math fournit un accès pratique aux constantes mathématiques les plus courantes et offre plusieurs fonctions mathématiques. Toutes les propriétés et méthodes de la classe Math sont statiques et doivent être appelées à l'aide de la syntaxe `Math.`*`méthode`*`(`*`paramètre`*`)` ou `Math.`*`constante`*. Reportez-vous à Math.
Microphone	La classe Microphone vous permet d'accéder au micro de l'utilisateur, s'il en a installé un. Utilisé avec Flash Media Server, votre fichier SWF peut diffuser et enregistrer les sons du microphone d'un utilisateur. Reportez-vous à Microphone.
Mouse	La classe Mouse vous permet de contrôler la souris dans un fichier SWF. Vous pouvez l'utiliser pour masquer ou afficher le pointeur de la souris, par exemple. Reportez-vous à Mouse.
MovieClip	Chaque clip d'un fichier SWF est une occurrence de la classe MovieClip. Utilisez les méthodes et propriétés de cette classe pour contrôler les objets de clip. Reportez-vous à MovieClip.

Classe	Description
MovieClipLoader	Cette classe permet d'implémenter des rappels d'écouteur qui fournissent des informations d'état lors du chargement des fichiers SWF, JPEG, GIF et PNG dans les occurrences de clips. Reportez-vous à MovieClipLoader.
NetConnection	La classe NetConnection établit une connexion locale en flux continu pour la lecture de fichiers Flash Video (FLV) à partir d'une adresse HTTP ou du système de fichiers local. Reportez-vous à NetConnection.
NetStream	La classe NetStream contrôle la lecture des fichiers FLV à partir d'un système de fichiers local ou d'une adresse HTTP. Reportez-vous à NetStream.
Number	La classe Number est une enveloppe pour le type de données primitif numérique. Reportez-vous à Number.
Object	La classe Object est à la racine de la hiérarchie des classes ActionScript. Toutes les autres classes héritent de ses méthodes et propriétés. Reportez-vous à Object.
PrintJob	La classe PrintJob permet d'imprimer du contenu à partir d'un fichier SWF, y compris le contenu créé de façon dynamique et les documents à plusieurs pages. Reportez-vous à PrintJob.
Selection	La classe Selection permet de définir et de contrôler le champ de texte dans lequel se trouve le point d'insertion (celui qui a le focus). Reportez-vous à Selection.
SharedObject	La classe SharedObject crée un stockage de données permanent sur l'ordinateur client, similaire aux cookies. Cette classe offre le partage des données en temps réel entre objets de l'ordinateur client. Reportez-vous à SharedObject.
Sound	La classe Sound vous permet de contrôler les sons dans un fichier SWF. Reportez-vous à Sound.
Stage	La classe Stage fournit des informations sur les dimensions, l'alignement et le mode d'échelle du fichier SWF. Elle signale également les événements de redimensionnement Stage. Reportez-vous à Stage.
String	La classe String est une enveloppe pour le type de données primitif chaîne, ce qui vous permet d'utiliser les méthodes et les propriétés de l'objet String pour manipuler les valeurs primitives de chaîne. Reportez-vous à String.

Classe	Description
System	La classe System fournit des informations sur Flash Player et sur le système sur lequel il s'exécute (la résolution d'affichage et la langue utilisée, par exemple). Elle vous permet aussi d'afficher ou de masquer le panneau Paramètres Flash Player et de modifier les paramètres de sécurité du fichier SWF Reportez-vous à System.
TextField	La classe TextField permet de contrôler les champs texte de saisie et dynamiques, notamment pour extraire des informations de formatage, appeler des gestionnaires d'événement et changer des propriétés telles que les valeurs alpha et la couleur d'arrière-plan. Reportez-vous à TextField.
TextFormat	La classe TextFormat vous permet d'appliquer des styles de formatage aux caractères et aux paragraphes d'un objet TextField. Reportez-vous à TextFormat.
TextSnapshot	L'objet TextSnapshot permet d'accéder au texte statique d'un clip et de mettre ce texte en forme. Reportez-vous à TextSnapshot.
Video	La classe Video permet d'afficher des objets vidéo dans un fichier SWF. Vous pouvez l'utiliser avec Flash Media Server pour afficher un contenu vidéo en flux continu dans un fichier SWF ou dans Flash pour afficher un fichier Flash Video (FLV). Reportez-vous à Video.
XML	Cette classe contient les méthodes et propriétés réservées aux objets XML. Reportez-vous à XML.
XMLNode	La classe XMLNode représente un nœud unique dans une arborescence de documents XML. Elle constitue la super-classe de la classe XML. Reportez-vous à XMLNode.
XMLSocket	La classe XMLSocket permet de créer une connexion socket durable entre un serveur et un client exécutant Flash Player. Les sockets du client autorisent un transfert de données à faible temps d'attente, tel que celui que requièrent les applications de discussion en ligne en temps réel (appelées « chat »). Reportez-vous à XMLSocket.
XMLUI	L'objet XMLUI autorise une communication avec les fichiers SWF utilisés comme interfaces utilisateur personnalisées pour les fonctions d'extensibilité de l'outil de programmation Flash (telles que Comportements, Commandes, Effets et Outils). Reportez-vous à XMLUI.

Le package flash.display

Le package flash.display contient la classe BitmapData que vous pouvez utiliser pour développer des affichages visuels.

Classe	Description
BitmapData	La classe BitmapData permet de créer des images bitmap transparentes ou opaques de tailles arbitraires ou des images dans le document et de les manipuler de diverses manières à l'exécution. Reportez-vous à BitmapData (flash.display.BitmapData).

Package flash.external

Le package flash.external permet de communiquer avec le conteneur Flash Player à l'aide de code ActionScript. Par exemple, si vous imbriquez un fichier SWF dans une page HTML, celle-ci devient le conteneur. Vous pouvez alors communiquer avec la page HTML via la classe ExternalInterface et JavaScript. Également appelé External API.

Classe	Description
ExternalInterface	La classe ExternalInterface est l'API External, un sous-système qui autorise les communications entre ActionScript et le conteneur Flash Player (par exemple une page HTML utilisant JavaScript) ou une application de bureau qui utilise Flash Player. Reportez-vous à ExternalInterface (flash.external.ExternalInterface).

Le package flash.filters

Le package flash.filters contient des classes liées aux effets de filtres bitmap disponibles dans Flash Player 8. Les filtres permettent d'appliquer de riches effets visuels, tels que le flou, les biseaux, le rayonnement et les ombres portées, aux occurrences d'image et de clip. Pour plus d'informations sur chaque classe, consultez les références fournies dans le tableau suivant.

Classe	Description
BevelFilter	*La classe BevelFilter permet d'ajouter un effet biseauté à une occurrence de clip.* Reportez-vous à BevelFilter (flash.filters.BevelFilter).
BitmapFilter	La classe BitmapFilter est une classe de base pour tous les effets de filtres. Reportez-vous à BitmapFilter (flash.filters.BitmapFilter).

Classe	Description
BlurFilter	*La classe* BlurFilter *permet d'appliquer un effet flou à des occurrences de clip.* Reportez-vous à BlurFilter (flash.filters.BlurFilter).
ColorMatrixFilter	La classe ColorMatrixFilter permet d'appliquer une transformation matricielle 4x5 aux valeurs de couleur RGBA et alpha de chaque pixel d'une image d'entrée. Après l'application de cette transformation, le résultat peut être produit avec un nouveau jeu de couleurs RGBA et de valeurs alpha. Reportez-vous à ColorMatrixFilter (flash.filters.ColorMatrixFilter).
ConvolutionFilter	La classe ConvolutionFilter permet d'appliquer un filtre de convolution de matrice Reportez-vous à ConvolutionFilter (flash.filters.ConvolutionFilter).
DisplacementMapFilter	La classe DisplacementMapFilter permet d'utiliser les valeurs des pixels d'une image spécifiée (remplacement d'une image) pour remplacer dans l'espace l'occurrence d'origine (un clip) auquel le filtre est appliqué. Reportez-vous à DisplacementMapFilter (flash.filters.DisplacementMapFilter).
DropShadowFilter	La classe DropShadowFilter permet d'ajouter un effet d'ombre portée à un clip. Reportez-vous à DropShadowFilter (flash.filters.DropShadowFilter).
GlowFilter	La classe GlowFilter permet d'ajouter un effet de rayonnement à un clip. Reportez-vous à GlowFilter (flash.filters.GlowFilter).
GradientBevelFilter	La classe GradientBevelFilter permet d'appliquer un effet dégradé à un clip. Reportez-vous à GradientBevelFilter (flash.filters.GradientBevelFilter).
GradientGlowFilter	La classe GradientGlowFilter permet d'appliquer un effet de rayonnement dégradé à un clip. Reportez-vous à GradientGlowFilter (flash.filters.GradientGlowFilter).

Package flash.geom

Le package flash.geom contient des classes de géométrie, telles que des points, des rectangles et des matrices de transformation. Ces classes prennent en charge la classe BitmapData et la fonctionnalité de mise en cache des bitmap. Pour plus d'informations sur chaque classe, consultez les références fournies dans le tableau suivant.

Classe	Description
ColorTransform	La classe ColorTransform permet de définir mathématiquement la valeur des couleurs RVB et de transformation de couleur d'une occurrence. Vous pouvez récupérer ces valeurs après leur définition. Reportez-vous à ColorTransform (flash.geom.ColorTransform).
Matrix	Représente une matrice de transformation qui détermine la façon de mapper des points d'un espace de coordonnées à l'autre. Reportez-vous à Matrix (flash.geom.Matrix).
Point	L'objet Point représente un emplacement dans un système de coordonnées à deux dimensions, dans lequel `x` est l'axe horizontal et `y` l'axe vertical. Reportez-vous à Point (flash.geom.Point).
Rectangle	La classe Rectangle permet de créer et de modifier les objets Rectangle. Reportez-vous à Rectangle (flash.geom.Rectangle).
Transform	Rassemble des données sur les transformations de couleurs et les manipulations de coordonnées qui s'appliquent à un objet occurrence. Reportez-vous à Transform (flash.geom.Transform).

Package flash.net

Le package flash.net contient des classes qui vous permettent de charger et télécharger un ou plusieurs fichiers entre l'ordinateur d'un utilisateur et le serveur. Pour plus d'informations sur chaque classe, consultez les références fournies dans le tableau suivant.

Classe	Description
FileReference	La classe FileReference vous permet de charger et télécharger un ou plusieurs fichiers entre l'ordinateur d'un utilisateur et le serveur. Reportez-vous à FileReference (flash.net.FileReference).
FileReferenceList	La classe FileReferenceList permet de charger un ou plusieurs fichiers entre l'ordinateur d'un utilisateur et le serveur. Reportez-vous à FileReferenceList (flash.net.FileReferenceList).

package flash.text

Le package flash.text comporte la classe TextRenderer permettant de travailler avec une fonctionnalité d'anti-crénelage avancée dans Flash Player8.

Classe	Description
TextRenderer	Cette classe fournit une fonctionnalité d'anti-crénelage avancé dans Flash Player8. Voir %{TextRenderer (flash.text.TextRenderer)}%..

Le package mx.lang

Le package mx.lang comporte la classe Locale pour travailler avec un texte multilingue.

Classe	Description
Locale	Cette classe vous permet de contrôler la façon dont le texte multilingue s'affiche dans un fichier SWF. Reportez-vous à Locale (mx.lang.Locale).

Les packages System et TextField

Le package System contient les capacités d'IME et des classes de sécurité. Ces classes traitent les paramètres du client susceptibles d'affecter le fonctionnement de votre application dans Flash Player. Pour plus d'informations sur chaque classe, consultez les références fournies dans le tableau suivant.

Classe	Description
capabilities	La classe Capabilities détermine les capacités du système et du Flash Player qui héberge le fichier SWF. Elle vous permet de personnaliser le contenu pour des formats différents. Reportez-vous à capabilities (System.capabilities).
IME	La classe IME permet de manipuler directement l'IME (Input Method Editor) du système d'exploitation sous lequel l'application Flash Player s'exécute sur l'ordinateur client. Reportez-vous à IME (System.IME).
security	La classe security contient des méthodes spécifiant la façon dont les fichiers SWF peuvent communiquer entre eux dans différents domaines. Reportez-vous à security (System.security).

Le package TextField contient la classe StyleSheet qui vous permet d'appliquer des styles CSS au texte.

Classe	Description
StyleSheet	La classe StyleSheet permet de créer un objet feuille de style contenant les règles de formatage du texte, comme la taille et la couleur de la police et d'autres styles de formatage. Reportez-vous à StyleSheet (TextField.StyleSheet).

Utilisation des classes intégrées

En programmation orientée objet (OOP), une *classe* définit une catégorie d'objets. Une classe décrit les propriétés (données) et le comportement (méthodes) d'un objet, comme un plan d'architecte décrit les caractéristiques d'un immeuble. Pour plus de détails sur les classes et autres concepts de programmation orientée objet, consultez les rubriques suivantes :

- « Bases de la programmation orientée objet. », à la page 204
- « Ecriture de fichiers de classe personnalisée », à la page 208

Flash comporte de nombreuses classes intégrées que vous pouvez utiliser dans votre code (voir « Présentation des classes de niveau supérieur et intégrées », à la page 263), ce qui vous aide à rajouter facilement de l'interactivité à vos applications. Pour utiliser les propriétés et méthodes définies par une classe, vous devez tout d'abord créer une *occurrence* de cette classe (sauf pour les classes comportant des membres statiques). Le rapport entre une occurrence et sa classe est similaire au rapport entre une maison et le plan d'architecte correspondant, comme évoqué dans « Présentation des classes de niveau supérieur et intégrées », à la page 263.

Pour plus d'informations sur l'utilisation des classes qui sont intégrées à Flash, consultez les rubriques suivantes :

- « Création d'une nouvelle occurrence d'une classe intégrée », à la page 275
- « Accès aux propriétés intégrées des objets », à la page 275
- « Appel de méthodes d'objet intégré », à la page 276
- « Présentation des membres (statiques) de classe », à la page 276
- « Préchargement de fichiers de classe », à la page 278
- « Exclusion de classes », à la page 277

Création d'une nouvelle occurrence d'une classe intégrée

Pour créer une occurrence de classe ActionScript, utilisez l'opérateur `new` pour appeler la fonction constructeur de la classe. La fonction constructeur porte toujours le même nom que la classe dont elle renvoie une occurrence, que vous attribuez généralement à une variable.

Le code suivant crée par exemple un nouvel objet Sound :

```
var song_sound:Sound = new Sound();
```

Dans certains cas, il n'est pas nécessaire de créer une occurrence de classe pour employer ses propriétés et ses méthodes. Pour plus d'informations, voir « Présentation des membres (statiques) de classe », à la page 276.

Accès aux propriétés intégrées des objets

Utilisez l'opérateur point (`.`) pour accéder à la valeur d'une propriété d'objet. Entrez le nom de l'objet à gauche du point et le nom de la propriété à droite. Par exemple, dans l'instruction suivante, `my_obj` représente l'objet et `firstName` la propriété :

```
my_obj.firstName
```

L'exemple de code suivant crée un nouvel objet Array puis affiche sa propriété `length` :

```
var my_array:Array = new Array("apples", "oranges", "bananas");
trace(my_array.length); // 3
```

Vous pouvez également utiliser l'opérateur d'accès tableau (`[]`) pour accéder aux propriétés d'un objet, notamment au cours des processus de débogage. L'exemple suivant passe en boucle sur un objet pour afficher toutes ses propriétés :

Pour passer en boucle les contenus d'un objet :

1. Créez un nouveau document Flash, puis enregistrez-le sous le nom **forin.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

   ```
   var results:Object = {firstName:"Tommy", lastName:"G", age:7, avg:0.336,
     b:"R", t:"L"};
   for (var i:String in results) {
       trace("the value of [" + i + "] is: " + results[i]);
   }
   ```

   Le code précédent définit un nouvel Objet appelé résultats et définit des valeurs pour `firstName`, `lastName`, `age`, `avg`, `b`, et `t`. Un `for..in` en boucle suit chaque propriété dans l'objet résultat et suit sa valeur au panneau de sortie.
3. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

Pour plus d'informations sur la création de tableaux et leurs opérateurs d'accès, reportez-vous à « Présentation des opérateurs », à la page 145. Pour plus d'informations sur les méthodes et les propriétés, consultez le Chapitre 5, « Fonctions et méthodes », à la page 171. Pour des exemples de l'utilisation des propriétés de la classe intégrée MovieClip, reportez-vous à Chapitre 10, « Utilisation des clips », à la page 335, pour des exemples de l'utilisation des propriétés des classes TextField, String, TextRenderer, et TextFormat, consultez le Chapitre 11, « Utilisation du texte et des chaînes », à la page 367.

Appel de méthodes d'objet intégré

Pour appeler la méthode d'un objet, utilisez l'opérateur point (`.`), suivi de la méthode. Par exemple, le code suivant crée un nouvel objet Sound et appelle sa méthode `setVolume()` :

```
var my_sound:Sound = new Sound(this);
my_sound.setVolume(50);
```

Pour des exemples de l'utilisation des méthodes de la classe intégrée MovieClip, voir Chapitre 10, « Utilisation des clips », à la page 335. Pour des exemples de l'utilisation des méthodes de la classe intégrée TextField, String, TextRenderer, et TextFormat, voir Chapitre 11, « Utilisation du texte et des chaînes », à la page 367.

Présentation des membres (statiques) de classe

Certaines classes ActionScript intégrées possèdent des *membres de classe* (*membres statiques*). Pour invoquer les membres de classe (propriétés et méthodes) ou y accéder, vous utilisez le nom de la classe, et non son occurrence. Vous ne devez donc pas créer d'occurrence de la classe pour utiliser ces propriétés et méthodes.

Par exemple, toutes les propriétés de la classe Math sont statiques. Le code suivant appelle la méthode `max()` de la classe Math pour déterminer lequel de deux nombres est le plus grand.

```
var largerNumber:Number = Math.max(10, 20);
trace(largerNumber); // 20
```

Pour plus d'informations sur les méthodes statiques de la classe Math, et des exemples d'utilisation, reportez-vous à Math dans le *Guide de référence du langage ActionScript 2.0*.

Exclusion de classes

Pour réduire la taille d'un fichier SWF, il peut être nécessaire d'exclure des classes de la compilation tout en restant capable d'y accéder et de les utiliser pour la vérification du type. Par exemple, procédez de cette façon lorsque vous développez une application qui utilise plusieurs fichiers SWF ou des bibliothèques partagées, en particulier celles qui accèdent souvent aux mêmes classes. Cette exclusion de classes vous aide à éviter la duplication de classes dans ces fichiers.

Pour plus d'informations sur l'utilisation des classes, consultez les rubriques suivantes :

- « Préchargement de fichiers de classe », à la page 278

Pour exclure des classes de la compilation :

1. Créez un nouveau fichier XML.
2. Nommez le fichier XML ***FLA_filename*_exclude.xml**, où *FLA_filename* correspond au nom du fichier FLA sans son extension.

   Par exemple, si le fichier FLA s'appelle sellStocks.fla, le nom de fichier XML doit être sellStocks_exclude.xml.
3. Enregistrez le fichier dans le même répertoire que le fichier FLA.
4. Placez les balises suivantes dans le fichier XML :

```
<excludeAssets>
  <asset name="className1" />
  <asset name="className2" />
</excludeAssets>
```

   Les valeurs spécifiées pour les attributs de nom dans les balises `<asset>` désignent les noms des classes à exclure du fichier SWF. Ajoutez-en autant que nécessaire pour votre application. Par exemple, le fichier XML suivant exclut les classes mx.core.UIObject et mx.screens.Slide du fichier SWF :

```
<excludeAssets>
  <asset name="mx.core.UIObject" />
  <asset name="mx.screens.Slide" />
</excludeAssets>
```

Pour plus d'informations sur les classes de préchargement, consultez la section « Préchargement de fichiers de classe », à la page 278.

Préchargement de fichiers de classe

Cette section décrit certaines méthodologies de préchargement et d'exportation des classes dans Flash (y compris les classes qu'utilisent les composants de la version 2 de l'architecture des composants). *Le préchargement* implique le chargement de certaines des données sur un fichier SWF avant que l'utilisateur ne commence à interagir. Flash importe les classes dans la première image d'un fichier SWF lorsque vous utilisez des classes externes. Ces données constituent le premier élément à charger dans un fichier SWF. Il est similaire aux classes de composant, dans la mesure où la structure des composants se charge également dans la première image d'un fichier SWF. Lorsque vous créez de grandes applications, le temps de chargement peut être considérable lorsque vous devez importer des données. Vous devez donc traiter ces données de façon plus technique, comme indiqué dans les procédures suivantes.

Dans la mesure où les classes constituent les premières données à charger, vous risquez d'avoir à créer des difficultés à créer une barre de progression ou à charger une animation lorsque les classes se chargent avant la barre de progression, car la barre de progression doit rendre compte de l'ensemble du processus de chargement, classes incluses. Par conséquent, il est recommandé de charger les classes après les autres parties du fichier SWF, mais avant d'utiliser les composants.

La procédure suivante indique comment modifier l'image dans laquelle les classes se chargent dans un fichier SWF.

Pour sélectionner une autre image pour les classes à charger dans le fichier SWF :

1. Choisissez Fichier > Paramètres de publication.
2. Sélectionnez l'onglet Flash, puis cliquez sur le bouton Paramètres.
3. Dans la zone de texte Exporter l'Image pour les Classes, tapez le numéro d'une nouvelle image, ce qui va déterminer quand charger les classes.
4. Cliquez sur OK.

Vous ne pouvez pas utiliser n'importe quelle classe tant que la tête de lecture n'a pas atteint l'image cible du chargement. Par exemple, les composants de la version 2 nécessitent des classes pour leur fonctionnalité ; vous devez donc charger des composants après l'image exportée pour les classes ActionScript 2.0. Si vous procédez à l'exportation pour l'image 3, vous ne pouvez pas exploiter ces classes tant que la tête de lecture n'a pas atteint l'image 3 et n'a pas chargé les données.

Si vous devez précharger un fichier utilisant des composants, par exemple les classes de composants de la version 2, vous devez les précharger dans le fichier SWF. Pour ce faire, vous devez définir vos composants de façon à les exporter pour une autre image dans le fichier SWF. Par défaut, les composants UI s'exportent vers l'image 1 du fichier SWF ; assurez-vous donc que vous avez désélectionné Export dans First Frame de la boîte de dialogue de liaison du composant.

REMARQUE	Pour ajouter un composant sur la scène à l'aide de code ActionScript, vous devez faire glisser une occurrence du composant à ajouter dans le panneau de collage (la zone entourant la scène). Ceci indique à Flash que vous utilisez le composant dans votre application, et qu'il ne s'agit pas d'un élement inutilisé de la bibliothèque. N'oubliez pas que Flash n'ajoute pas d'éléments inutilisés de la bibliothèque aux fichiers SWF.

Lorsque les composants ne se chargent pas sur la première image, vous pouvez créer une barre de progression personnalisée pour la première image du fichier SWF. Ne référencez aucun composant dans votre code ActionScript ou n'incluez aucun composant sur la scène pendant le chargement des classes destinées à l'image spécifiée dans le champ Export Frame for Classes.

ATTENTION	Vous devez exporter les composants après les classes ActionsScript qu'ils utilisent.

CHAPITRE 7

Héritage

7

Dans le Chapitre 6, « Classes », vous avez appris à écrire des fichiers de classe et à utiliser les classes pour organiser le code dans des fichiers externes. Ce chapitre décrivait également l'organisation des fichiers de classe dans des packages associés. L'objectif de ce chapitre est de vous apprendre à écrire des classes plus complexes, qui étendent les fonctionnalités d'une classe existante. Ce sujet est intéressant car vous serez probablement amené à étendre vos propres classes personnalisées ou les classes existantes pour pouvoir ajouter des nouvelles méthodes et des propriétés.

Pour plus d'informations sur les héritages, consultez la section « Présentation de l'héritage », à la page 281. Pour plus d'informations sur les méthodes et les propriétés, consultez le Chapitre 5, « Fonctions et méthodes », à la page 171.

Pour plus d'informations sur l'héritage, consultez les sections suivantes :

Présentation de l'héritage

Vous avez découvert dans le Chapitre 6, « Classes » comment créer un fichier de classe pour créer vos propres types de données personnalisés. A travers ces fichiers de classe personnalisée, vous avez également appris à extraire le code du scénario pour le placer dans des fichiers externes. Ce déplacement vers des fichiers externes simplifie l'édition de votre code. Maintenant que les bases de la création de classes personnalisées n'ont plus de secrets pour vous, vous allez découvrir une technique de programmation orientée objet appelée *sous-classement* ou *extension de classe*, qui vous permet de créer de nouvelles classes à partir d'une classe existante.

L'un des avantages de la programmation orientée objet réside dans la possibilité de créer des *sous-classes* d'une classe. Ces sous-classes héritent alors de toutes les propriétés et méthodes de la *super-classe*. Par exemple, si vous étendez (ou *sous-classez*) la classe MovieClip, vous créez une classe personnalisée qui étend la classe MovieClip. Votre sous-classe hérite donc de toutes les propriétés et méthodes de la classe MovieClip. Vous pouvez également créer un ensemble de classes étendant une super-classe personnalisée. La classe Laitue, par exemple, peut être une extension de la super-classe Légume.

Votre sous-classe définit généralement d'autres méthodes et propriétés utilisables dans votre application et, en cela, *étend* la super-classe. Les sous-classes peuvent également supplanter (apporter leurs propres définitions) les méthodes héritées d'une super-classe. Lorsqu'une sous-classe remplace une méthode héritée de sa super-classe, la définition de la superclasse n'est plus accessible depuis la sous-classe. La seule exception à cette règle est la suivante : si vous êtes dans la fonction constructeur d'une sous-classe, vous pouvez appeler la fonction de constructeur de la superclasse à l'aide de l'instruction `super`. Pour plus d'informations sur le remplacement de méthodes et de propriétés, consultez la section « Remplacement des méthodes et des propriétés », à la page 287.

Par exemple, vous créez une classe Mammifère qui définit certaines propriétés et certains comportements communs à tous les mammifères. Vous pouvez alors créer une sous-classe Chat qui permet d'étendre la classe Mammifère. L'utilisation de sous-classes permet de recycler le code de sorte que, au lieu de recréer le code commun à deux classes, il vous suffit d'étendre une classe existante. Une autre sous-classe peut étendre la classe Chat (la classe Siamois par exemple), et ainsi de suite. Dans une application complexe, la définition de la structure hiérarchique des classes représente une grande partie du processus de création.

L'héritage et le sous-classement se révèlent très utiles dans les grandes applications car ils vous permettent de créer une série de classes associées partageant les mêmes fonctionnalités. Par exemple, vous pouvez créer une classe Employé définissant les méthodes et les propriétés de base d'un employé typique dans une société. Vous pouvez ensuite créer une nouvelle classe appelée Fournisseur qui étend la classe Employé et hérite de toutes ses méthodes et propriétés. La classe Fournisseur pourrait également ajouter ses propres méthodes et propriétés ou supplanter celles de la super-classe Employé. Vous pouvez même créer une classe Dirigeant étendant également la classe Employé et définissant d'autres méthodes et propriétés telles que `recrutement()`, `licenciement()`, `augmentation()` et `promotion()`. Vous pouvez encore étendre une sous-classe, par exemple la classe Dirigeant, et créer une nouvelle classe appelée Directeur, ajoutant à son tour de nouvelles méthodes ou remplaçant les méthodes existantes.

Chaque fois que vous étendez une classe existante, la nouvelle classe hérite de toutes les méthodes et propriétés de la sous-classe. Si aucune des classes n'était associée, il vous faudrait réécrire chaque méthode et propriété dans chaque fichier de classe, même lorsque la fonctionnalité ne change pas. Vous perdriez alors du temps, non seulement dans l'écriture du code, mais également dans le débogage de votre application et la maintenance du projet lorsque le même code doit être modifié dans plusieurs fichiers.

Dans ActionScript, utilisez le mot-clé `extends` pour établir une relation d'héritage entre une classe et sa super-classe, ou pour étendre une interface. Pour plus d'informations sur l'utilisation du mot-clé`extends`, consultez les sections « Ecriture de sous-classes dans Flash », à la page 283 et « Ecriture d'une sous-classe », à la page 284. Pour plus d'informations sur le mot-clé `extends`, consultez la section relative à l'instruction extends dans le *Guide de référence du langage ActionScript 2.0*.

Ecriture de sous-classes dans Flash

En programmation orientée objet, une sous-classe peut hériter des propriétés et méthodes d'une autre classe, appelée *super-classe*. Vous pouvez étendre vos propres classes personnalisées et la plupart des classes de base et des classes ActionScript de Flash Player. Vous ne pouvez pas étendre la classe TextField.

Pour créer ce type de relation entre deux classes, utilisez la clause `extends` de l'instruction class. Pour spécifier une super-classe, utilisez la syntaxe suivante :

```
class SubClass extends SuperClass {}
```

La classe spécifiée dans SubClass hérite de toutes les propriétés et méthodes définies dans la super-classe.

Par exemple, vous créez une classe Mammifère qui définit des propriétés et méthodes communes à tous les mammifères. Pour créer une variante de cette classe Mammifère (Mammal), telle que la classe Marsupial, étendez la classe Mammal, c'est-à-dire créez une sous-classe de la classe Mammal, comme suit :

```
class Marsupial extends Mammal {}
```

La sous-classe hérite de toutes les propriétés et méthodes de la superclasse, y compris des propriétés ou des méthodes que vous avez déclarées comme étant privées en utilisant le mot-clé `private`.

Pour plus d'informations sur l'extension des classes, consultez les rubriques suivantes :

- « Ecriture d'une sous-classe », à la page 284
- « Remplacement des méthodes et des propriétés », à la page 287

Pour plus d'informations sur les membres privés, consultez la section « Présentation des méthodes et propriétés (membres) publiques, privées et statiques », à la page 221. Pour un exemple de création de sous-classe, consultez la section « Exemple : Extension de la classe Widget », à la page 285.

Ecriture d'une sous-classe

Le code suivant définit la classe personnalisée JukeBox, qui étend la classe Sound (Son). Il définit un tableau appelé `song_arr` et une méthode appelée `playSong()` qui permet de lire une chanson et d'invoquer la méthode `loadSound()`, dont il hérite de la classe Sound.

```
class JukeBox extends Sound {
  public var song_arr:Array = new Array("beethoven.mp3", "bach.mp3",
  "mozart.mp3");
  public function playSong(songID:Number):Void {
    super.loadSound(song_arr[songID], true);
  }
}
```

Si vous ne placez pas un appel à `super()` dans la fonction constructeur d'une sous-classe, le compilateur génère automatiquement un appel au constructeur de sa superclasse immédiate, sans paramètre, en tant que première instruction de la fonction. Si la super-classe n'a pas de constructeur, le compilateur crée une fonction constructeur vide, puis génère un appel à cette fonction à partir de la sous-classe. Cependant, si une super-classe prend des paramètres dans sa définition, vous devez créer un constructeur dans la sous-classe et appeler la super-classe avec les paramètres requis.

L'héritage multiple, ou héritage à partir de plusieurs classes, n'est pas autorisé dans ActionScript 2.0. Toutefois, les classes peuvent effectivement hériter de plusieurs classes, si vous utilisez des instructions `extends` individuelles, comme dans l'exemple suivant :

```
// non autorisé
class C extends A, B {} // **Erreur : Une classe ne peut pas étendre
                        // plusieurs classes.

// autorisé
class B extends A {}
class C extends B {}
```

Vous pouvez également utiliser des interfaces pour mettre en oeuvre une forme limitée d'héritage multiple. Pour plus d'informations sur les interfaces, consultez le Chapitre 8, « Interfaces », à la page 295. Pour un exemple de création de sous-classe, consultez la section « Exemple : Extension de la classe Widget », à la page 285. Pour plus d'informations sur `super`, consultez la section relative à l'instruction super dans le *Guide de référence du langage ActionScript 2.0*.

Exemple : Extension de la classe Widget

Les membres de classe se propagent dans les sous-classes de la super-classe qui définit ces membres. L'exemple suivant décrit la création d'une classe Widget, que vous étendez (sous-classe) via l'écriture d'une sous-classe SubWidget.

Pour créer la classe Widget et la sous-classe SubWidget :

1. Créez un fichier ActionScript et enregistrez-le sous le nom **Widget.as**.
2. Ajoutez le code suivant dans le nouveau document :

```
class Widget {
  public static var widgetCount:Number = 0;
  public function Widget() {
    Widget.widgetCount++;
  }
}
```

3. Enregistrez les modifications apportées au fichier ActionScript.
4. Créez un fichier et enregistrez-le sous le nom **SubWidget.as** dans le même répertoire que la classe Widget.
5. Dans SubWidget.as, saisissez le code suivant dans la fenêtre de script :

```
class SubWidget extends Widget {
  public function SubWidget() {
    trace("Creating subwidget #" + Widget.widgetCount);
  }
}
```

6. Enregistrez les modifications apportées à SubWidget.as.
7. Créez un fichier FLA et enregistrez-le en tant que **subWidgetTest.fla** dans le même répertoire que les fichiers de classe ActionScript précédents.
8. Dans le fichier subWidgetTest.fla, saisissez le code suivant dans l'image 1 du scénario principal :

```
var sw1:SubWidget = new SubWidget();
var sw2:SubWidget = new SubWidget();
trace("Widget.widgetCount = " + Widget.widgetCount);
trace("SubWidget.widgetCount = " + SubWidget.widgetCount);
```

Le code précédent crée deux occurrences de la classe SubWidget : sw1 et sw2. Chaque appel au constructeur du `SubWidget` affiche la valeur actuelle de la propriété statique de `Widget.widgetCount`. Comme la classe SubWidget est une sous classe de la classe Widget, vous pouvez accéder à la propriété `widgetCount` par l'intermédiaire de la classe SubWidget, et le compilateur écrit de nouveau la référence (dans le pseudo-code binaire, pas dans votre fichier ActionScript) sous la forme `Widget.widgetCount`. Si vous tentez d'accéder à la propété statique `widgetCount` par des occurrences de la classe Widget ou SubWidget, comme sw1 ou sw2, le compilateur émet une erreur.

9. Enregistrez les modifications apportées au document.
10. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

    Le panneau de sortie affiche les données de sortie suivantes :

```
Creating subwidget #1
Creating subwidget #2
Widget.widgetCount = 2
SubWidget.widgetCount = 2
```

    Ce résultat est dû au fait que, bien que le constructeur de la classe Widget ne soit jamais appelé explicitement, le constructeur de la classe SubWidget l'appelle à votre place.
    Le constructeur de la classe Widget incrémente donc la variable statique `widgetCount` de la classe Widget.

    Le compilateur d'ActionScript 2.0 peut résoudre des références à des membres statiques au sein des définitions de classe.

    Si vous ne spécifiez pas le nom de classe de la propriété `Widget.widgetCount` et la référencez simplement avec `widgetCount`, le compilateur d'ActionScript 2.0 la résout en `Widget.widgetCount` et exporte correctement cette propriété. De même, si vous faites référence à la propriété en tant que `SubWidget.widgetCount`, le compilateur écrit de nouveau la référence (dans le pseudo-code binaire, pas dans votre fichier ActionScript) sous la forme `Widget.widgetCount` car SubWidget est une sous-classe de la classe Widget.

ATTENTION	Si vous tentez d'accéder à la variable statique `widgetCount` à partir de la classe Widget à l'aide des occurrences sw1 ou sw2, Flash génère une erreur qui vous indique que les membres statiques ne peuvent être accédés que via les classes.

Pour améliorer la lisibilité du code, Adobe recommande de toujours employer des références explicites aux variables de membres statiques, comme dans l'exemple précédent.
Ces références explicites facilitent la localisation des définitions des membres statiques.

Remplacement des méthodes et des propriétés

Lorsqu'une sous-classe étend une super-classe, la sous-classe hérite de toutes les méthodes et propriétés de cette super-classe. L'un des avantages de l'utilisation et de l'extension des classes réside non seulement dans la possibilité d'apporter une nouvelle fonctionnalité à une classe existante, mais également de modifier la fonctionnalité existante. Prenons par exemple la classe Widget créée à la section « Exemple : Extension de la classe Widget », à la page 285. Vous pouvez maintenant créer une nouvelle méthode dans votre super-classe (Widget), puis supplanter la méthode de votre sous-classe (SubWidget), ou simplement utiliser la méthode héritée de la classe Widget. L'exemple suivant montre comment supplanter les méthodes existantes dans vos classes.

Pour supplanter les méthodes d'une sous-classe :

1. Créez un document ActionScript, puis enregistrez-le sous le nom **Widget.as**.
2. Dans Widget.as, saisissez le code ActionScript suivant dans la fenêtre de script :

   Note: Si vous avez créé la classe Widget dans un exemple précédent, modifiez le code existant en ajoutant la méthode `doSomething()`, comme suit :

```
class Widget {
  public static var widgetCount:Number = 0;
  public function Widget() {
    Widget.widgetCount++;
  }
  public function doSomething():Void {
    trace("Widget::doSomething()");
  }
}
```

3. Enregistrez les modifications apportées au document ActionScript.

   La classe Widget définit maintenant un constructeur et une méthode publique appelée `doSomething()`.

4. Créez un fichier ActionScript nommé **SubWidget.as** et enregistrez-le dans le même répertoire que Widget.as.

REMARQUE	Si vous avez créé la classe SubWidget à la section « Exemple : Extension de la classe Widget », à la page 285, vous pouvez utiliser ce fichier à la place.

5. Dans SubWidget.as, saisissez le code ActionScript suivant dans la fenêtre de script :

```
class SubWidget extends Widget {
  public function SubWidget() {
    trace("Creating subwidget # " + Widget.widgetCount);
    doSomething();
  }
}
```

6. Enregistrez les modifications apportées à SubWidget.as.

   Notez que le constructeur de la classe SubWidget appelle la méthode `doSomething()` que vous avez définie dans la super-classe.

7. Créez un document Flash et enregistrez-le sous **subWidgetTest.fla** dans le même répertoire que les documents ActionScript.

8. Dans subWidgetTest.fla, saisissez le code ActionScript suivant sur l'image 1 du scénario principal :

```
var sw1:SubWidget = new SubWidget();
var sw2:SubWidget = new SubWidget();
```

9. Enregistrez les modifications apportées au document Flash.

10. Choisissez Contrôle > Tester l'animation pour tester le document Flash. Les données de sortie suivantes doivent s'afficher dans le panneau de sortie :

```
Creating subwidget # 1
Widget::doSomething()
Creating subwidget # 2
Widget::doSomething()
```

    Ce résultat indique que le constructeur de la classe SubWidget appelle le constructeur de sa super-classe (Widget), qui incrémente la propriété statique `widgetCount`.
    Le constructeur de SubWidget suit la propriété statique de la super-classe et appelle la méthode `doSomething()`, qui hérite de la super-classe.

11. Ouvrez la classe SubWidget et ajoutez une nouvelle méthode appelée `doSomething()`. Modifiez votre classe de sorte qu'elle corresponde au code suivant (ajoutez le code présenté en gras) :

```
class SubWidget extends Widget {
  public function SubWidget() {
    trace("Creating subwidget # " + Widget.widgetCount);
    doSomething();
  }
  public function doSomething():Void {
    trace("SubWidget::doSomething()");
  }
}
```

12. Enregistrez les modifications apportées au fichier de classe, puis ouvrez de nouveau subwidgetTest.fla.

13. Choisissez Contrôle > Tester l'animation pour tester le fichier. Les données de sortie suivantes doivent s'afficher dans le panneau de sortie :

```
Creating subwidget # 1
SubWidget::doSomething()
Creating subwidget # 2
SubWidget::doSomething()
```

Le résultat précédent montre que la méthode `doSomething()` du constructeur de la classe SubWidget appelle la méthode `doSomething()` de la classe active et non celle de la super-classe.

Ouvrez de nouveau la classe SubWidget, et modifiez le constructeur de la classe SubWidget pour appeler la méthode `doSomething()` de la super-classe (ajoutez le code présenté en gras) :

```
public function SubWidget() {
  trace("Creating subwidget # " + Widget.widgetCount);
  super.doSomething();
}
```

Comme nous l'avons démontré, vous pouvez ajouter le mot-clé `super` pour appeler la méthode `doSomething()` de la super-classe plutôt que la méthode `doSomething()` de la classe active. Pour plus d'informations sur le mot-clé `super`, consultez la section correspondante dans le manuel Guide de référence du langage ActionScript 2.0.

14. Enregistrez le fichier de la classe SubWidget et le constructeur modifié, puis choisissez Contrôle > Tester l'animation pour republier le document Flash.

Le panneau de sortie affiche le contenu de la méthode `doSomething()` de la classe Widget.

Utilisation du polymorphisme dans une application

La programmation orientée objet permet d'exprimer les différences entre les classes individuelles par une technique appelée polymorphisme. Cette technique permet aux classes de supplanter les méthodes de leurs super-classes et de définir des implémentations spécialisées de ces méthodes.

Par exemple, vous pouvez commencer par une classe appelée Mammal qui comporte les méthodes `play()` et `sleep()`. Vous créez ensuite les sous-classes Chat, Singe et Chien pour étendre la classe Mammifère. Les sous-classes supplantent la méthode `play()` de la classe Mammal, de façon à représenter de façon plus réaliste ces types d'animaux. La classe Monkey implémente la méthode `play()` pour se balancer aux arbres ; la classe Cat implémente la méthode `play()` pour courir après une pelote ; la classe Dog implémente la méthode `play()` pour rapporter une balle. Dans la mesure où la fonctionnalité `sleep()` reste similaire quel que soit l'animal, vous utiliseriez l'implémentation de super-classe. La procédure suivante montre cet exemple dans Flash.

Pour utiliser le polymorphisme dans une application :

1. Créez un document ActionScript et enregistrez-le sous le nom **Mammal.as**.

   Ce document est la classe de base des quelques classes d'animaux différentes que vous allez créer dans les étapes suivantes.

2. Dans Mammal.as, saisissez le code ActionScript suivant dans la fenêtre de script :

```
class Mammal {
  private var _gender:String;
  private var _name:String = "Mammal";

  // Constructeur
  public function Mammal(gender:String) {
    this._gender = gender;
  }

  public function toString():String {
    return "[object " + speciesName + "]";
  }
  public function play():String {
    return "Chase another of my kind.";
  }
  public function sleep():String {
    return "Close eyes.";
  }
```

```
    public function get gender():String {
      return this._gender;
    }
    public function get speciesName():String {
      return this._name;
    }
    public function set speciesName(value:String):Void {
      this._name = value;
    }
}
```

La classe précédente définit deux variables privées, `_gender` et `_name`, servant à stocker le type de mammifère et le sexe de l'animal. Le constructeur de `Mammal` est ensuite défini. Ce constructeur prend un seul paramètre, `gender`, utilisé pour définir la variable privée `_gender` définie précédemment. Trois méthodes publiques supplémentaires sont également spécifiées : `toString()`, `play()`, et `sleep()`, chacune renvoyant des objets de chaîne. Les trois méthodes finales sont des méthodes de lecture et de définition pour les propriétés `_gender` et `_name` de Mammal.

3. Enregistrez le document ActionScript.

   Cette classe sert de super-classe pour les classes Cat, Dog et Monkey, que vous créerez par la suite. Vous pouvez utiliser la méthode `toString()` de la classe Mammal pour afficher une représentation de chaîne de n'importe quelle occurrence de Mammal (ou toute occurrence qui étendait la classe Mammal).

4. Créez un fichier ActionScript et enregistrez-le sous le nom **Cat.as** dans le même répertoire que le fichier de la classe Mammal.as créé à l'étape 1.

5. Dans Cat.as, saisissez le code ActionScript suivant dans la fenêtre de script :

```
class Cat extends Mammal {
  // Constructeur
  public function Cat(gender:String) {
    super(gender);
    speciesName = "Cat";
  }

  public function play():String {
    return "Pounce a ball of yarn.";
  }
}
```

Notez que vous supplantez la méthode `play()` de la super-classe Mammal. La classe Cat ne définit que deux méthodes : une méthode constructeur et une méthode `play()`. Comme la classe Cat étend la classe Mammal, elle hérite de ses méthodes et de ses propriétés. Pour plus d'informations sur le remplacement de méthodes et de propriétés, consultez la section « Remplacement des méthodes et des propriétés », à la page 287.

6. Enregistrez les modifications apportées au document ActionScript.
7. Créez un document et enregistrez-le sous le nom **Dog.as** dans le même répertoire que les deux fichiers de classe précédents.
8. Dans Dog.as, saisissez le code ActionScript suivant dans la fenêtre de script :

```
class Dog extends Mammal {
  // Constructeur
  public function Dog(gender:String) {
    super(gender);
    speciesName = "Dog";
  }

  public function play():String {
    return "Fetch a stick.";
  }
}
```

   Remarquez que la structure de la classe Dog est très similaire à celle de la classe Cat, seules quelques valeurs changent. Là encore, la classe Dog étend la classe Mammal et hérite de ses méthodes et propriétés. Le constructeur Dog prend une seule propriété, gender, qu'il transmet à la classe parent de la classe Dog, Mammal. La variable `speciesName` est également remplacée et définie sur la chaîne `Dog`. La méthode `play()` de la classe parent est également supplantée.
9. Enregistrez les modifications apportées au document ActionScript.
10. Créez un autre document ActionScript dans le même répertoire que vos autres fichiers, puis enregistrez-le sous le nom **Monkey.as**.
11. Dans Monkey.as, saisissez le code ActionScript suivant dans la fenêtre de script :

```
class Monkey extends Mammal {
  // Constructeur
  public function Monkey(gender:String) {
    super(gender);
    speciesName = "Monkey";
  }

  public function play():String {
    return "Swing from a tree.";
  }
}
```

   Comme les deux classes précédentes, Cat et Dog, la classe Monkey étend la classe Mammal. Le constructeur de cette classe appelle le constructeur de la classe Mammal, en transmettant le genre au constructeur de Mammal, et en définissant `speciesName` sur la chaîne `Monkey`. La classe Monkey supplante également le comportement de la méthode `play()`.

12. Enregistrez les modifications apportées au document ActionScript.
13. Maintenant que vous avez créé trois sous-classes de la classe Mammal, créez un document Flash appelé **mammalTest.fla**.
14. Dans mammalTest.fla, saisissez le code ActionScript suivant dans l'image 1 du scénario principal :

```
var mammals_arr:Array = new Array();
this.createTextField("info_txt", 10, 10, 10, 450, 80);
info_txt.html = true;
info_txt.multiline = true;
info_txt.border = true;
info_txt.wordWrap = true;

createMammals()
createReport()

function createMammals():Void {
  mammals_arr.push(new Dog("Female"));
  mammals_arr.push(new Cat("Male"));
  mammals_arr.push(new Monkey("Female"));
  mammals_arr.push(new Mammal("Male"));
}

function createReport():Void {
  var i:Number;
  var len:Number = mammals_arr.length;
  // Affichage des infos relatives à Mammal en 4 colonnes de texte HTML
  // et arrêts de tabulation.
  info_txt.htmlText = "<textformat tabstops='[110, 200, 300]'>";
  info_txt.htmlText += "<b>Mammal\tGender\tSleep\tPlay</b>";
  for (i = 0; i < len; i++) {
    info_txt.htmlText += "<p>" + mammals_arr[i].speciesName
        + "\t" + mammals_arr[i].gender
        + "\t" + mammals_arr[i].sleep()
        + "\t" + mammals_arr[i].play() + "</p>";
    // L'instruction trace appelle la méthode Mammal.toString().
    trace(mammals_arr[i]);
  }
  info_txt.htmlText += "</textformat>";
}
```

Le code de mammalTest.fla est un peu plus complexe que celui des classes précédentes. Il commence par importer trois classes d'animaux.

15. Enregistrez le document Flash, puis choisissez Contrôle > Tester l'animation pour tester le document.

    Les informations relatives à la classe Mammal s'affichent dans un champ de texte sur la scène, et le texte suivant apparaît dans le panneau de sortie :

```
[object Dog]
[object Cat]
[object Monkey]
[object Mammal]
```

CHAPITRE 8

Interfaces

8

En programmation orientée objet, une interface est un document qui vous permet de déclarer (mais pas de définir) les méthodes devant apparaître au sein d'une classe. Si vous faites partie d'une équipe de développeurs, ou si vous développez de grandes applications dans Flash, les interfaces vous seront certainement très utiles. Grâce à elles, les développeurs peuvent facilement identifier les méthodes de base dans les classes ActionScript. Ces méthodes doivent être implémentées lorsque les développeurs utilisent chaque interface.

Ce chapitre présente quelques exemples d'interfaces et, à la fin de sa lecture, vous serez à même de développer vos propres fichiers d'interface. Si vous ne maîtrisez pas le développement des classes, commencez par lire le Chapitre 6, « Classes », avant d'effectuer les didacticiels et d'examiner les exemples de ce chapitre.

Pour plus d'informations sur l'utilisation des interfaces, consultez les rubriques suivantes :

Présentation des interfaces

En programmation orientée objet, les interfaces sont comme des classes dont les méthodes ne sont pas implémentées (définies), c'est-à-dire que, mis à part cela, elles ne « font » rien. Une interface se compose donc de méthodes « vides ». Une autre classe peut ensuite implémenter les méthodes déclarées par l'interface. Dans ActionScript, la distinction entre interface et objet porte sur la détection des erreurs lors de la compilation et l'application des règles du langage.

Une interface n'est pas une classe. Cependant, ceci n'est pas tout à fait vrai dans ActionScript lors de l'exécution du code car une interface est une abstraction. Il est possible d'utiliser des interfaces ActionScript pendant l'exécution pour effectuer une attribution de type (attribuer un autre type de données à des données existantes). Le modèle d'objet ActionScript 2.0 ne prend pas en charge l'héritage multiple. Par conséquent, une classe ne peut hériter que d'une classe parent. Cette classe parent peut être soit de base, soit une classe Flash Player, soit une classe définie par l'utilisateur (personnalisée). Vous pouvez utiliser les interfaces pour appliquer une forme limitée d'héritage multiple, dans laquelle la classe hérite de plusieurs classes.

Par exemple, en C++, la classe Chat peut étendre la classe Mammal, ainsi que la classe Playful, qui a les méthodes `chaseTail()` et `eatCatNip()`. Comme Java, ActionScript 2.0 ne permet pas d'étendre directement plusieurs classes, mais permet d'étendre une classe unique et d'implémenter plusieurs interfaces. Par conséquent, vous pouvez créer une interface Playful qui déclare les méthodes `chaseTail()` et `eatCatNip()`. Une classe Chat ou toute autre classe peut alors implémenter cette interface et fournir des définitions pour ces méthodes.

Une interface peut également être considérée comme un « contrat de programmation » destiné à imposer des relations entre des classes non connexes. Par exemple, supposons que vous travailliez avec une équipe de programmeurs et que chacun de vous travaille sur une classe différente de la même application. Lors de la réalisation de l'application, vous vous mettez d'accord sur un ensemble de méthodes que les différentes classes utiliseront pour communiquer. Ainsi, vous créez une interface qui déclare ces méthodes, leurs paramètres et leurs types de renvoi. Toute classe qui implémente cette interface doit fournir des définitions pour ces méthodes ; dans le cas contraire, une erreur du compilateur se produit. L'interface se comporte comme un protocole de communication auquel toutes les classes doivent adhérer.

Pour ce faire, vous pouvez créer une classe qui définit toutes ces méthodes, puis faire en sorte que chaque classe étende cette super-classe, ou en hérite. Cependant, l'application étant composée de classes n'ayant aucun rapport entre elles, il est inutile de toutes les placer dans une hiérarchie de classes communes. Il est préférable de créer une interface qui déclare les méthodes que ces classes utilisent pour communiquer, puis que chaque classe implémente ces méthodes (en fournissant ses propres définitions).

Vous pouvez généralement programmer de façon efficace sans utiliser les interfaces. Lorsqu'elles sont utilisées de façon appropriée, cependant, les interfaces peuvent rendre la conception de vos applications plus élégante, évolutive et stable.

Les interfaces d'ActionScript existent pendant l'exécution pour autoriser l'attribution de type. Reportez-vous à Chapitre 3, « Attribution des objets », à la page 78 Une interface n'est ni objet ni une classe, mais son flux de travaux est le même que celui des classes. Pour plus d'informations sur le flux de travaux des classes, consultez la section « Ecriture de fichiers de classe personnalisée », à la page 208. Un didacticiel sur la création d'une application avec interfaces est à votre disposition à la section « Exemple : Utilisation des interfaces », à la page 305.

Pour plus d'informations sur l'utilisation des interfaces, consultez les sections suivantes :

Présentation du mot-clé interface

Le mot-clé `interface` définit une interface. Une interface est similaire à une classe. Les différences fondamentales sont regroupées ci-dessous :

- Les interfaces contiennent uniquement les déclarations des méthodes, pas leur implémentation. Ainsi, toute classe qui implémente une interface doit fournir une implémentation pour chaque méthode déclarée dans l'interface.
- Seuls les membres publics sont autorisés dans la définition d'une interface. Les membres statiques et les membres de classe ne sont pas permis.
- Les instructions `get` et `set` ne sont pas autorisées dans les définitions d'interface.
- Pour utiliser le mot-clé `interface`, vous devez spécifier ActionScript 2.0 et Flash Player 6 ou une version plus récente dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA.

Le mot-clé `interface` n'est pris en charge que lorsqu'il est utilisé dans des fichiers de script externes, et non dans les scripts écrits dans le panneau Actions.

Affectation des noms d'interfaces

Les noms d'interface doivent commencer par une majuscule, comme les noms de classe. Les noms d'interface sont généralement des adjectifs, tels que `Imprimable`. Le nom d'interface suivant, `IEmployeeRecords`, commence par une majuscule et le mot concaténé commence par une majuscule suivie de minuscules :

```
interface IEmployeeRecords {}
```

REMARQUE

Certains développeurs font débuter les noms d'interface par un I majuscule pour les distinguer des classes. Cette méthode est judicieuse car elle permet de distinguer rapidement les interfaces et les classes ordinaires.

Pour plus d'informations sur les conventions d'appellation, consultez le Chapitre 17, « Recommandations et conventions de programmation pour ActionScript 2.0 », à la page 715.

Définition et implémentation des interfaces

Le processus de création d'une interface est identique au processus de création d'une classe. Comme pour les classes, vous pouvez définir des interfaces uniquement dans des fichiers ActionScript externes. Le flux de travaux de création d'une interface implique au minimum les étapes suivantes :

- Définition d'une interface dans un fichier ActionScript externe.
- Enregistrement du fichier d'interface dans un répertoire de chemin de classe spécifique (emplacement où Flash recherche les classes) ou dans le même répertoire que le fichier FLA de l'application
- Création d'une occurrence de la classe dans un autre script, soit dans un document Flash (FLA), soit dans un fichier de script externe, soit dans des sous-interfaces basées sur l'interface d'origine.
- Création d'une classe qui implémente l'interface dans un fichier de script externe

Déclarez une interface à l'aide du mot-clé `interface`, suivi du nom de l'interface et d'accolades gauche et droite (`{}`), qui définissent le corps de l'interface, comme dans l'exemple suivant :

```
interface IEmployeeRecords {
  // déclarations de méthodes d'interface
}
```

Une interface ne peut contenir que des déclarations de méthodes (fonction), y compris des paramètres, des types de paramètres et des types de renvoi de fonction.

Pour plus d'informations sur les conventions de structuration des classes et interfaces, consultez le Chapitre 17, « Recommandations et conventions de programmation pour ActionScript 2.0 », à la page 715. Un didacticiel sur la création d'une application utilisant une interface est à votre disposition à la section « Exemple : Utilisation des interfaces », à la page 305.

Par exemple, le code suivant déclare une interface nommée `IMyInterface` qui contient deux méthodes, `method1()` et `method2()`. La première méthode, `method_1()`, ne comporte pas de paramètre et spécifie le type Void (ce qui signifie qu'aucune valeur n'est renvoyée).
La deuxième méthode, `method2()`, ne prend qu'un seul paramètre de type String et spécifie un renvoi de type booléen.

Pour créer une interface simple :

1. Créez un fichier ActionScript et enregistrez-le sous le nom d'**IMyInterface.as**.
2. Saisissez le code ActionScript suivant dans la fenêtre de script :

```
interface IMyInterface {
  public function method1():Void;
  public function method2(param:String):Boolean;
}
```

3. Enregistrez les modifications apportées au fichier ActionScript.

   Pour utiliser la nouvelle interface dans une application, vous devez d'abord créer une classe qui l'implémente.
4. Créez un fichier ActionScript et enregistrez-le sous le nom de **MyClass.as** dans le même répertoire que IMyInterface.as.
5. Dans le fichier de classe MyClass, saisissez le code ActionScript suivant dans la fenêtre de script :

```
class MyClass {
}
```

   Pour demander à la classe personnalisée (MyClass) d'utiliser votre interface (IMyInterface), utilisez le mot-clé `implements`qui spécifie qu'une classe doit définir toutes les méthodes déclarées dans l'interface (ou les interfaces) que vous implémentez.
6. Modifiez le code ActionScript dans le fichier MyClass.as (ajoutez le code présenté en gras) de sorte qu'il corresponde au code suivant :

```
class MyClass implements IMyInterface {
}
```

   Placez le mot-clé `implements` après le nom de la classe.
7. Cliquez sur le bouton Vérifier la syntaxe.

Flash affiche une erreur dans le panneau de sortie pour signaler que MyClass doit implémenter la méthode *X* de l'interface IMyInterface. Ce message d'erreur apparaît car toute classe qui étend une interface doit définir chaque méthode énumérée dans le document de l'interface.

8. Modifiez de nouveau le document MyClass (ajoutez le code présenté en gras) et écrivez le code ActionScript pour les méthodes `method1()` et `method2()`, comme dans l'exemple suivant :

```
class MyClass implements IMyInterface {
  public function method1():Void {
    // ...
  };
  public function method2(param:String):Boolean {
    // ...
    return true;
  }
}
```

9. Enregistrez le document MyClass.as et cliquez sur Vérifier la syntaxe.

   Le panneau de sortie n'affiche plus de message d'erreur ou d'avertissement car les deux méthodes sont à présent définies.

Le fichier de classe que vous avez créé n'est pas limité aux méthodes publiques définies dans le fichier d'interface. Le fichier d'interface ne présente que les méthodes devant être implémentées au minimum, avec leurs propriétés et leurs types de retour. Les classes qui implémentent une interface particulière incluent presque toujours d'autres méthodes, variables et méthodes de lecture et de définition.

Les fichiers d'interfaces ne peuvent contenir aucune déclaration ni affectation de variable. Les fonctions déclarées dans une interface ne doivent pas contenir d'accolades. Par exemple, l'interface suivante ne sera pas compilée :

```
interface IBadInterface {
  // Erreur de compilation. Les déclarations de variables ne sont
  // pas autorisées dans les interfaces.
  public var illegalVar:String;

  // Erreur de compilation. Les corps de fonctions ne sont pas autorisés
  // dans les interfaces.
  public function illegalMethod():Void {
  }

  // Erreur de compilation. Les méthodes privées sont interdites dans
  // les interfaces.
  private function illegalPrivateMethod():Void;

  // Erreur de compilation. Les méthodes de lecture/définition sont
  // interdites dans les interfaces.
  public function get illegalGetter():String;
}
```

Un didacticiel sur la création d'une interface complexe est à votre disposition à la section « Exemple : Utilisation des interfaces », à la page 305.

Les règles d'affectation de nom et de stockage des interfaces dans les paquets sont les mêmes que celles des classes ; consultez « Appellation des fichiers de classe », à la page 241.

Création d'interfaces comme types de données

Tout comme une classe, une interface définit un nouveau type de données. Toute classe qui implémente une interface peut être considérée comme relevant du type défini par l'interface. Ceci est utile pour déterminer si un objet donné implémente une interface donnée. Prenons par exemple l'interface `IMovable` que vous allez créer dans l'exemple suivant.

Pour créer une interface en tant que type de données :

1. Créez un document ActionScript et enregistrez-le sur votre disque dur sous le nom d'**IMovable.as**.
2. Dans IMovable.as, saisissez le code ActionScript suivant dans la fenêtre de script :

```
interface IMovable {
  public function moveUp():Void;
  public function moveDown():Void;
}
```

3. Enregistrez les modifications apportées au fichier ActionScript.
4. Créez un document ActionScript et enregistrez-le sous le nom de **Box.as** dans le même répertoire que IMovable.as.

   Dans ce document, vous créez une classe Box implémentant l'interface IMovable créée précédemment.
5. Dans Box.as, tapez le code ActionScript suivant dans la fenêtre de script :

```
class Box implements IMovable {
  public var xPos:Number;
  public var yPos:Number;

  public function Box() {
  }

  public function moveUp():Void {
    trace("moving up");
    // définition de méthode
  }
  public function moveDown():Void {
    trace("moving down");
    // définition de méthode
  }
}
```

6. Enregistrez les modifications apportées au document ActionScript.
7. Créez un document Flash nommé **boxTest.fla,** et enregistrez-le dans le même répertoire que les documents ActionScript précédents.
8. Sélectionnez l'image 1 du scénario, ouvrez l'éditeur ActionScript, puis tapez le code suivant dans le panneau Actions :

   ```
   var newBox:Box = new Box();
   ```

   Ce code ActionScript crée une occurrence de la classe Box, que vous allez déclarer en tant que variable du type Box.
9. Enregistrez les modifications apportées au document Flash, puis choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

   A l'exécution, dans Flash Player7 et ses versions ultérieures, vous pouvez attribuer une expression à un type d'interface ou à un autre type de données lors de l'exécution. Contrairement aux interfaces Java, des interfaces ActionScript existent lors de l'exécution du script, ce qui autorise l'attribution de type. Si l'expression est un objet qui implémente l'interface, ou si elle possède une super-classe qui implémente l'interface, l'objet est renvoyé. Dans le cas contraire, `null` est renvoyé. Cela est particulièrement utile si vous souhaitez vous assurer qu'un objet particulier implémente une interface particulière. Pour plus d'informations sur l'attribution de type, consultez le Chapitre 3, « Attribution des objets », à la page 78.
10. Dans boxTest.fla, ajoutez le code suivant à la fin du code ActionScript :

    ```
    if (IMovable(newBox) != null) {
      newBox.moveUp();
    } else {
      trace("box instance is not movable");
    }
    ```

    Ce code ActionScript contrôle si l'occurrence `newBox` implémente l'interface IMovable avant l'appel de la méthode `moveUp()` sur l'objet.
11. Enregistrez le document Flash, puis sélectionnez Contrôle > Tester l'animation pour tester le fichier SWF.

    Comme l'occurrence Box implémente l'interface IMovable, la méthode `Box.moveUp()` est appelée et le texte « moving up » s'affiche dans le panneau de sortie.

Pour plus d'informations sur l'attribution, consultez le Chapitre 3, « Attribution des objets », à la page 78.

Fonctionnement des héritages et des interfaces

Le mot-clé `extends` vous permet de créer des sous-classes d'une interface. Cela peut se révéler très utile pour les vastes projets dans lesquels vous souhaitez étendre (ou *sous-classer*) une interface existante et ajouter d'autres méthodes. Ces méthodes doivent être définies pour toutes les classes qui implémentent cette interface.

Lorsque vous étendez des interfaces, vous devez savoir que vous recevez des messages d'erreur dans Flash si plusieurs fichiers d'interface déclarent des fonctions portant le même nom mais ayant des paramètres ou des types de retour différents.

L'exemple suivant présente le sous-classement d'un fichier d'interface à l'aide du mot-clé `extends`.

Pour étendre une interface :

1. Créez un fichier ActionScript et enregistrez-le sous le nom de **Ia.as**.
2. Dans Ia.as, saisissez le code ActionScript suivant dans la fenêtre de script :

```
interface Ia {
  public function f1():Void;
  public function f2():Void;
}
```

3. Enregistrez les modifications apportées au fichier ActionScript.
4. Créez un fichier ActionScript et enregistrez-le sous le nom de **Ib.as**, dans le même répertoire que le fichier Ia.as créé à l'étape 1.
5. Dans Ib.as, saisissez le code ActionScript suivant dans la fenêtre de script :

```
interface Ib extends Ia {
  public function f8():Void;
  public function f9():Void;
}
```

6. Enregistrez les modifications apportées au fichier ActionScript.
7. Créez un fichier ActionScript et enregistrez-le sous le nom de **ClassA.as** dans le même répertoire que les deux fichiers précédents.

8. Dans ClassA.as, saisissez le code ActionScript suivant dans la fenêtre de script :

```
class ClassA implements Ib {
  // f1() et f2() sont définies dans l'interface Ia.
  public function f1():Void {
  }
  public function f2():Void {
  }

  // f8() et f9() sont définies dans l'interface Ib qui étend Ia.
  public function f8():Void {
  }
  public function f9():Void {
  }
}
```

9. Enregistrez votre fichier de classe, puis cliquez sur le bouton Vérifier la syntaxe sous la fenêtre de script.

   Flash ne génère pas de message d'erreur tant que les quatre méthodes sont définies et correspondent aux définitions provenant de leurs fichiers d'interface respectifs.

REMARQUE	Dans ActionScript2.0, chaque classe ne peut étendre qu'une classe, même si vous pouvez utiliser les classes pour implémenter autant d'interfaces que nécessaire.

Pour que votre classe ClassA implémente plusieurs interfaces dans l'exemple précédent, il vous suffit de séparer les interfaces par des virgules. Ou, si l'une de vos classes étend une super-classe et implémente plusieurs interfaces, servez-vous d'un code semblable au suivant :

```
class ClassA extends ClassB implements Ib, Ic, Id {...}.
```

Exemple : Utilisation des interfaces

Dans cet exemple, vous allez créer une interface simple que vous pourrez ensuite réutiliser entre plusieurs classes différentes.

Pour créer une interface :

1. Créez un fichier ActionScript et enregistrez-le sous le nom de **IDocumentation.as**.
2. Dans IDocumentation.as, saisissez le code ActionScript suivant dans la fenêtre de script :

```
interface IDocumentation {
  public function downloadUpdates():Void;
  public function checkForUpdates():Boolean;
  public function searchHelp(keyword:String):Array;
}
```

3. Enregistrez les modifications apportées au fichier d'interface ActionScript.
4. Créez un fichier ActionScript dans le même répertoire que IDocumentation.as, puis enregistrez-le sous le nom de **FlashPaper.as.**
5. Dans FlashPaper.as, saisissez le code ActionScript suivant dans la fenêtre de script :

```
class FlashPaper implements IDocumentation {
}
```

6. Enregistrez les modifications apportées au fichier ActionScript.
7. Cliquez sur le bouton Vérifier la syntaxe pour votre classe ActionScript.

   Un message semblable au suivant apparaît :

```
**Error** path\FlashPaper.as: Line 1: The class must implement method
  'checkForUpdates' from interface 'IDocumentation'.

     class FlashPaper implements IDocumentation {

Total ActionScript Errors: 1  Reported Errors: 1
```

   Cette erreur est due au fait que la classe FlashPaper active ne définit aucune des méthodes publiques que vous avez définies dans l'interface IDocumentation.

8. Ouvrez de nouveau le fichier de classe FlashPaper.as et modifiez le code ActionScript existant de sorte qu'il corresponde au suivant :

```
class FlashPaper implements IDocumentation {
  private static var __version:String = "1,2,3,4";
  public function downloadUpdates():Void {
  };
  public function checkForUpdates():Boolean {
    return true;
  };
  public function searchHelp(keyword:String):Array {
    return []
  };
}
```

9. Enregistrez les modifications apportées au fichier ActionScript et cliquez de nouveau sur le bouton Vérifier la syntaxe.

   Aucun message ne s'affiche plus dans le panneau de sortie.

> **REMARQUE** Vous pouvez ajouter autant de méthodes ou variables statiques, publiques ou privées supplémentaires que nécessaire dans le fichier de classe FlashPaper. Le fichier d'interface ne définit qu'un ensemble minimum de méthodes devant apparaître au sein des classes qui implémentent cette interface.

10. Ouvrez de nouveau le document d'interface IDocumentation et ajoutez la ligne de code en gras suivante (au-dessous de la méthode `searchHelp()`) :

```
interface IDocumentation {
  public function downloadUpdates():Void;
  public function checkForUpdates():Boolean;
  public function searchHelp(keyword:String):Array;
  public function addComment(username:String, comment:String):Void;
}
```

11. Enregistrez les modifications apportées au fichier d'interface et ouvrez de nouveau le document FlashPaper.as.

12. Cliquez sur le bouton Vérifier la syntaxe et un nouveau message d'erreur apparaît dans le panneau de sortie :

```
**Error** path\FlashPaper.as: Line 1: The class must implement method
  'addComment' from interface 'IDocumentation'.

     class FlashPaper implements IDocumentation {

Total ActionScript Errors: 1  Reported Errors: 1
```

   L'erreur précédente est due au fait que le fichier de classe FlashPaper.as ne définit plus toutes les classes décrites dans le fichier d'interface. Pour corriger l'erreur, vous devez soit ajouter la méthode `addComment()` à la classe FlashPaper, soit supprimer la définition de la méthode du fichier d'interface IDocumentation.

13. Ajoutez la méthode suivante dans la classe FlashPaper :

```
public function addComment(username:String, comment:String):Void {
  /* Envoi de paramètres à la page côté serveur, qui insère un
  commentaire dans la base de données. */
}
```

14. Enregistrez les modifications apportées à FlashPaper.as, puis cliquez sur le bouton Vérifier la syntaxe. Aucune erreur ne devrait plus apparaître.

Dans la section précédente, vous avez créé une classe basée sur le fichier d'interface IDocumentation. Dans cette section, vous allez créer une nouvelle classe qui implémente également l'interface IDocumentation, tout en ajoutant d'autres méthodes et propriétés.

Ce didacticiel démontre l'utilité des interfaces car si vous souhaitez créer une autre classe étendant l'interface IDocumentation, vous pouvez aisément identifier les méthodes nécessaires au sein de la nouvelle classe.

Exemple : Création d'une interface complexe

L'exemple suivant présente plusieurs façons de définir et d'implémenter des interfaces. Dans ce didacticiel, vous allez apprendre à créer un fichier d'interface simple, à écrire une classe qui implémente plusieurs interfaces et à utiliser des interfaces qui étendent d'autres interfaces pour créer des structures de données plus complexes.

Pour créer une interface complexe :

1. Créez un document ActionScript, puis enregistrez-le sous le nom d'**InterfaceA.as**.
2. Créez un dossier appelé **complexInterface** et enregistrez InterfaceA.as dans ce répertoire.

   Vous enregistrerez ensuite tous les fichiers créés pour ce didacticiel dans ce répertoire.
3. Dans Interface.as, saisissez le code ActionScript suivant dans la fenêtre de script :

```
// Nom de fichier : InterfaceA.as
interface InterfaceA {
  public function k():Number;
  public function n(z:Number):Number;
}
```

4. Enregistrez le document ActionScript, puis créez-en un autre appelé **ClassB.as** et enregistrez-le dans le répertoire complexInterface.

   ClassB.as implémente l'interface InterfaceA créée précédemment.
5. Dans ClassB.as, saisissez le code ActionScript suivant dans la fenêtre de script :

```
// Nom de fichier : ClassB.as
class ClassB implements InterfaceA {
  public function k():Number {
    return 25;
  }
  public function n(z:Number):Number {
    return (z + 5);
  }
}
```

6. Enregistrez les modifications apportées au document ClassB.as, puis créez un autre document Flash nommé **classbTest.fla** dans le répertoire complexInterface.

   Ce fichier de classe teste la classe ClassB créée précédemment.

7. Dans classbTest.fla, saisissez le code ActionScript suivant dans l'image 1 du scénario :

```
// Nom de fichier : classbTest.fla
import ClassB;
var myB:ClassB = new ClassB();
trace(myB.k()); // 25
trace(myB.n(7)); // 12
```

8. Enregistrez les modifications apportées au document Flash, puis choisissez Contrôle > Tester l'animation pour tester le document Flash.

   Le panneau de sortie affiche deux nombres, 25 et 12, résultats des méthodes `k()` et `n()` de la classe ClassB.

9. Créez un fichier ActionScript et enregistrez-le sous **ClassC.as** dans le répertoire complexInterface.

   Ce fichier de classe implémente l'interface InterfaceA créée à l'étape 1.

10. Dans ClassC.as, saisissez le code ActionScript suivant dans la fenêtre de script :

```
// Nom de fichier : ClassC.as
class ClassC implements InterfaceA {
  public function k():Number {
    return 25;
  }
  // **Erreur** La classe doit implémenter la méthode 'n' depuis
  // l'interface 'InterfaceA'.
}
```

    Si vous cliquez sur le bouton Vérifier la syntaxe pour le fichier de classe ClassC, Flash affiche un message d'erreur dans le panneau de sortie signalant que la classe active doit implémenter la méthode `n()` définie dans l'interface InterfaceA. Lorsque vous créez des classes qui implémentent une interface, il est important de définir des méthodes pour chaque entrée de l'interface.

11. Créez un document ActionScript et enregistrez-le sous le nom d'**InterfaceB.as** dans le répertoire complexInterface.

12. Dans InterfaceB.as, saisissez le code ActionScript suivant dans la fenêtre de script :

```
// Nom de fichier : InterfaceB.as
interface InterfaceB {
  public function o():Void;
}
```

13. Enregistrez les modifications apportées au document InterfaceB.as, puis créez un autre document ActionScript sous le nom de **ClassD.as** dans le répertoire complexInterface.

    Cette classe implémente les interfaces InterfaceA et InterfaceB créées précédemment. La classe ClassD doit inclure des implémentations de méthode pour chaque méthode énumérée dans chaque fichier d'interface.

14. Dans ClassD.as, saisissez le code ActionScript suivant dans la fenêtre de script :

```
// Nom de fichier : ClassD.as
class ClassD implements InterfaceA, InterfaceB {
  public function k():Number {
    return 15;
  }
  public function n(z:Number):Number {
    return (z * z);
  }
  public function o():Void {
    trace("o");
  }
}
```

15. Enregistrez les modifications apportées au fichier ClassD.as, puis créez un document Flash et enregistrez-le sous le nom de **classdTest.fla**.

    Ce document Flash teste la classe ClassD créée précédemment.

16. Dans classdTest.fla, saisissez le code ActionScript suivant dans l'image 1 du scénario :

```
// Nom de fichier : classdTest.fla
import ClassD;
var myD:ClassD = new ClassD();
trace(myD.k()); // 15
trace(myD.n(7)); // 49
myD.o(); // o
```

17. Enregistrez les modifications apportées au fichier classdTest.fla, puis choisissez Contrôle > Tester l'animation pour tester le fichier.

    Les valeurs 15 et 49 et la lettre o doivent apparaître dans le panneau de sortie. Ces valeurs sont les résultats respectifs des méthodes `ClassD.k()`, `ClassD.n()` et `ClassD.o()`.

18. Créez un document ActionScript, puis enregistrez-le sous le nom d'**InterfaceC.as**.

    Cette interface étend l'interface InterfaceA créée précédemment, puis ajoute une nouvelle définition de méthode.

19. Dans InterfaceC.as, saisissez le code ActionScript suivant dans la fenêtre de script :

```
// Nom de fichier : InterfaceC.as
interface InterfaceC extends InterfaceA {
  public function p():Void;
}
```

20. Enregistrez les modifications apportées au fichier ActionScript, puis créez-en un autre et enregistrez-le sous le nom de **ClassE.as** dans le répertoire complexInterface.

    Cette classe implémente deux interfaces, InterfaceB et InterfaceC.

21. Dans ClassE.as, saisissez le code ActionScript suivant dans la fenêtre de script :

```
// Nom de fichier : ClassE.as
class ClassE implements InterfaceB, InterfaceC {
  public function k():Number {
    return 15;
  }
  public function n(z:Number):Number {
    return (z + 5);
  }
  public function o():Void {
    trace("o");
  }
  public function p():Void {
    trace("p");
  }
}
```

22. Enregistrez les modifications apportées au document ActionScript, puis créez un autre document Flash nommé **classeTest.fla** dans le répertoire complexInterface.

23. Dans classeTest.fla, saisissez le code ActionScript suivant dans l'image 1 du scénario :

```
// Nom de fichier : classeTest.fla
import ClassE;
var myE:ClassE = new ClassE();
trace(myE.k()); // 15
trace(myE.n(7)); // 12
myE.o(); // o
myE.p(); // p
```

24. Enregistrez le document Flash, puis sélectionnez Contrôle > Tester l'animation pour tester le fichier SWF.

Les valeurs `15`, `12`, `o` et `p` apparaissent dans le panneau de sortie. Ces valeurs sont renvoyées par les méthodes `ClassE.k()`, `ClassE.n()`, `ClassE.o()` et `ClassE.p()`. La classe ClassE implémentant les interfaces InterfaceB et InterfaceC, chaque méthode issue des deux fichiers d'interface doit être définie. Bien que les interfaces InterfaceB et InterfaceC ne définissent que les méthodes `o()` et `p()`, InterfaceC étend InterfaceA. Cela signifie que chacune des méthodes définies, `k()` et `n()`, doit également être implémentée.

CHAPITRE 9

Gestion d'événements

9

Les *événements* sont des actions qui se produisent lors de la lecture d'un fichier SWF. Un événement tel qu'un clic de souris ou une pression sur une touche est appelé *événement utilisateur* puisqu'il résulte d'une interaction directe avec l'utilisateur. Un événement généré automatiquement par Flash Player, tel que l'apparence initiale d'un clip sur la scène, est appelé *événement système* car il n'est pas généré directement par l'utilisateur.

Pour que votre application réagisse à des événements, vous devez utiliser des *gestionnaires d'événement* (code ActionScript associé à un objet et à un événement particuliers). Par exemple, lorsqu'un utilisateur clique sur un bouton sur la scène, la tête de lecture peut passer à l'image suivante. De même, à l'issue du chargement d'un fichier XML sur le réseau, le contenu du fichier peut s'afficher dans un champ de texte.

Vous pouvez gérer les événements avec ActionScript de plusieurs façons :

- « ActionScript et les événements », à la page 312
- « Utilisation des écouteurs d'événement », à la page 316
- « Association de gestionnaires d'événement à des boutons et des clips », à la page 321, plus précisément, gestionnaire on et gestionnaire onClipEvent.
- « Diffusion d'événements à partir d'occurrences de composant », à la page 325

L'utilisation de gestionnaires d'événement avec loadMovie (méthode MovieClip.loadMovie) peut donner des résultats imprévisibles. Si vous liez un gestionnaire d'événement à un bouton avec `on()` ou si vous créez un gestionnaire dynamique avec une méthode telle que onPress (gestionnaire MovieClip.onPress), puis que vous appelez `loadMovie()`, le gestionnaire d'événement ne sera plus disponible après le chargement du nouveau contenu. Cependant, si vous utilisez gestionnaire onClipEvent ou gestionnaire on pour lier un gestionnaire d'événement à un clip, puis que vous appelez `loadMovie()` pour ce clip, le gestionnaire d'événement restera disponible après le chargement du nouveau contenu.

Pour plus d'informations sur la gestion des événements, consultez les sections suivantes :

ActionScript et les événements

Dans Flash, le code ActionScript est exécuté lorsque survient un événement : par exemple, lors du chargement d'un clip, de la saisie d'une image-clé sur le scénario ou lorsque l'utilisateur clique sur un bouton. Ces événements peuvent être déclenchés par l'utilisateur ou par le système. Les utilisateurs cliquent avec leur souris et appuient sur des touches ; le système déclenche des événements lorsque certaines conditions sont remplies ou lorsque des traitements sont terminés (le chargement d'un fichier SWF, le scénario atteint une certaine image, la fin du téléchargement d'une image, etc).

Lorsqu'un événement survient, vous écrivez un *gestionnaire d'événement* pour y répondre par une action. Il est important de bien comprendre le moment et l'emplacement où surviennent les événements pour mieux identifier comment et où y répondre par une action et quels outils ActionScript utiliser dans chaque cas.

Les événements peuvent être regroupés en un certain nombre de catégories : les événements de souris et de clavier, qui surviennent lorsqu'un utilisateur interagit avec votre application Flash par l'intermédiaire de la souris et du clavier ; les événements de clip qui surviennent dans des clips et les événements d'image qui surviennent dans les images du scénario.

Evénements de souris et de clavier

Un utilisateur qui interagit avec votre application ou fichier SWF déclenche des événements avec sa souris et son clavier. Par exemple, lorsque l'utilisateur survole un bouton, l'événement `Button.onRollOver` ou `on(rollOver)` survient. Lorsqu'il clique sur un bouton, il déclenche l'événement `Button.onRelease`. S'il appuie sur une touche du clavier, l'événement `on (keyPress)` survient. Vous pouvez écrire du code sur une image ou joindre des scripts à une occurrence pour traiter ces événements et ajouter toute l'interactivité nécessaire.

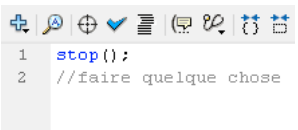
Evénements de clip

Au sein d'un clip, il est possible de réagir à un certain nombre d'événements déclenchés lorsque l'utilisateur entre ou quitte la scène ou interagit avec elle par le biais du clavier ou de la souris. Vous pouvez, par exemple, charger un fichier JPG ou SWF externe dans le clip lorsque l'utilisateur entre dans la scène. Il est également possible de permettre à l'utilisateur de déplacer les éléments de la scène par des mouvements de la souris.

Evénements d'image

Dans un scénario principal ou de clip, un événement système survient lorsque la tête de lecture arrive dans une image-clé. Cette opération est appelée *événement d'image*.
Les événements d'image sont très utiles pour déclencher des actions en fonction de l'écoulement du temps (progression dans le scénario) ou pour interagir avec les éléments actuellement visibles sur la scène. Lorsque vous ajoutez un script à une image-clé, il s'exécute dès que la lecture atteint cette image-clé. Un script joint à une image est appelé *script d'image*.

L'une des utilisations les plus courantes des scripts d'image consiste à arrêter la lecture dès qu'une certaine image-clé est atteinte. Cette opération s'effectue avec la fonction `stop()`. Vous sélectionnez une image-clé, puis ajoutez la fonction `stop()` en tant qu'élément de script dans le panneau Actions.



Après avoir arrêté le fichier SWF à une certaine image-clé, vous devez effectuer certaines actions. Vous pouvez par exemple utiliser un script d'image pour mettre à jour dynamiquement la valeur d'un intitulé, pour gérer l'interaction entre les éléments de la scène, etc.

Utilisation de méthodes de gestionnaire d'événement

Une méthode de gestionnaire d'événement est une méthode de classe invoquée lorsqu'un événement se produit sur une occurrence de cette classe. Par exemple, la classe MovieClip définit un gestionnaire d'événement `onPress`, invoqué lorsque l'utilisateur clique avec le bouton de la souris sur un objet clip. Contrairement aux autres méthodes de classe, vous n'invoquez pas directement un gestionnaire d'événement ; Flash Player le fait lui-même lorsque l'événement concerné se produit.

Les classes ActionScript suivantes sont des exemples de classes qui définissent des gestionnaires d'événement : Button, ContextMenu, ContextMenuItem, Key, LoadVars, LocalConnection, Mouse, MovieClip, MovieClipLoader, Selection, SharedObject, Sound, Stage, TextField, XML et XMLSocket. Pour plus d'informations sur les gestionnaires d'événement qu'elles proposent, consultez les entrées correspondantes dans le *Guide de référence du langage ActionScript 2.0*. Le terme *gestionnaire* est ajouté au titre de chaque gestionnaire d'événement.

Par défaut, les méthodes de gestionnaire d'événement ne sont pas définies : lorsqu'un événement particulier se produit, son gestionnaire associé est invoqué, mais votre application ne répond pas davantage à l'événement. Pour qu'elle réponde à l'événement, définissez une fonction au moyen de l'instruction function et affectez-la au gestionnaire d'événement approprié. La fonction que vous avez affectée au gestionnaire d'événement est ensuite invoquée automatiquement lorsque l'événement se produit.

Un gestionnaire d'événement se compose de trois éléments : l'objet auquel l'événement s'applique, le nom de la méthode du gestionnaire d'événement de l'objet et la fonction que vous avez affectée au gestionnaire d'événement. L'exemple suivant illustre la structure de base d'un gestionnaire d'événement :

```
object.eventMethod = function () {
  // Votre code, répondant à l'événement.
}
```

Par exemple, supposons que vous ayez un bouton next_btn sur la scène. Le code suivant affecte au gestionnaire d'événement `onPress` du bouton une fonction qui fait avancer la tête de lecture jusqu'à l'image suivante dans le scénario actuel :

```
next_btn.onPress = function () {
  nextFrame();
}
```

Affectation d'une référence de fonction Dans le code précédent, la fonction `nextFrame()` a été affectée à un gestionnaire d'événement pour `onPress`. Vous pouvez également affecter une référence de fonction (nom) à une méthode de gestionnaire d'événement et définir la fonction ultérieurement, comme illustré par l'exemple suivant.

```
// Affecte une référence de fonction au gestionnaire d'événement onPress
// du bouton.
next_btn.onPress = goNextFrame;

// Définit la fonction goNextFrame().
function goNextFrame() {
  nextFrame();
}
```

Notez que, dans l'exemple suivant, vous affectez la référence de fonction, et non la valeur renvoyée, au gestionnaire d'événement `onPress` :

```
// Incorrect !
next_btn.onPress = goNextFrame();
// Correct.
next_btn.onPress = goNextFrame;
```

Réception des paramètres transmis Certains gestionnaires d'événement reçoivent des paramètres qui fournissent des informations sur l'événement qui s'est produit. Par exemple, le gestionnaire d'événement `TextField.onSetFocus` est invoqué lorsqu'une occurrence de champ de texte parvient au focus clavier. Ce gestionnaire d'événement reçoit une référence à l'objet de champ de texte qui avait précédemment le focus clavier.

Par exemple, le code suivant insère du texte dans le champ de texte qui vient juste de perdre le focus clavier :

```
this.createTextField("my_txt", 99, 10, 10, 200, 20);
my_txt.border = true;
my_txt.type = "input";
this.createTextField("myOther_txt", 100, 10, 50, 200, 20);
myOther_txt.border = true;
myOther_txt.type = "input";
myOther_txt.onSetFocus = function(my_txt:TextField) {
  my_txt.text = "I just lost keyboard focus";
};
```

Gestionnaires d'événement pour les objets d'exécution Vous pouvez également affecter des fonctions aux gestionnaires d'événement pour les objets créés lors de l'exécution. Le code suivant, par exemple, crée une nouvelle occurrence de clip (`newclip_mc`) et affecte une fonction au gestionnaire d'événement `onPress` du clip :

```
this.attachMovie("symbolID", "newclip_mc", 10);
newclip_mc.onPress = function () {
  trace("You pressed me");
}
```

Pour plus d'informations, voir « Création de clips à l'exécution », à la page 344.

Contournement des méthodes de gestionnaire d'événement En créant une classe qui étend une classe ActionScript, vous pouvez supplanter les méthodes du gestionnaire d'événement avec les fonctions que vous avez écrites. Vous pouvez définir un gestionnaire d'événement dans une nouvelle sous-classe que vous pouvez utiliser de nouveau pour différents objets en liant les symboles de la bibliothèque de la classe étendue à la nouvelle sous-classe. Le code suivant contourne le gestionnaire d'événement `onPress` de la classe MovieClip par l'intermédiaire d'une fonction qui réduit la transparence du clip :

```
// Classe FadeAlpha - définit la transparence lorsque vous cliquez
// sur le clip.
class FadeAlpha extends MovieClip {
  function onPress() {
    this._alpha -= 10;
  }
}
```

Pour plus d'informations sur l'extension d'une classe ActionScript et sur la liaison avec les symboles de la bibliothèque, consultez les exemples de la section « Affectation d'une classe à des symboles dans Flash », à la page 256. Pour plus d'informations sur l'écriture et l'utilisation de classes personnalisées, consultez le Chapitre 6, « Classes »

Utilisation des écouteurs d'événement

Les écouteurs d'événement permettent à un objet, appelé *objet écouteur*, de recevoir des événements diffusés par un autre objet, appelé *objet diffuseur*. L'objet diffuseur enregistre l'objet écouteur afin de recevoir des événements générés par le diffuseur. Par exemple, vous enregistrez un objet de clip pour recevoir des notifications `onResize` de la scène, ou une occurrence de bouton peut recevoir des notifications `onChanged` d'un objet de champ de texte. Vous pouvez enregistrer plusieurs objets écouteurs pour recevoir des événements d'un seul diffuseur et vous pouvez enregistrer un seul objet écouteur pour recevoir des événements de plusieurs diffuseurs.

Contrairement aux méthodes de gestion d'erreurs, le modèle écouteur-diffuseur applicable aux événements permet d'utiliser plusieurs éléments de code pour écouter un même événement sans créer de conflits. Les modèles d'événement qui n'appliquent pas ce modèle, tels que `XML.onLoad`, peuvent devenir une source de problèmes lorsque plusieurs blocs de code écoutent un même événement. Ces différents blocs entrent en conflit pour contrôler la seule référence de la fonction de rappel `XML.onLoad`. Avec le modèle écouteur/diffuseur, vous pouvez ajouter facilement des écouteurs à un même événement sans avoir à vous soucier des blocages.

Les classes ActionScript suivantes permettent de diffuser des événements : Key, Mouse, MovieClipLoader, Selection, Stage et TextField. Pour identifier les écouteurs disponibles pour une classe, consultez chaque entrée de classe dans *Guide de référence du langage ActionScript 2.0*.

Pour plus d'informations sur les écouteurs d'événement, consultez les rubriques suivantes :

- « Modèle d'écouteur d'événement », à la page 317
- « Exemple d'écouteur d'événement », à la page 318

Pour un exemple de fichier source, stagesize.fla, qui décrit comment la propriété `Stage.scaleMode` affecte les valeurs de `Stage.width` et `Stage.height` lorsque la fenêtre du navigateur est redimensionnée, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/StageSize afin d'accéder à l'exemple.

Modèle d'écouteur d'événement

Le modèle des écouteurs d'événement est similaire à celui des gestionnaires d'événement (consultez la section « ActionScript et les événements », à la page 312), à deux différences près :

- Vous affectez le gestionnaire d'événement à l'objet écouteur et non pas à l'objet qui diffuse l'événement.
- Vous appelez une méthode spéciale de l'objet diffuseur, `addListener()`, qui enregistre l'objet écouteur pour recevoir ses événements.

Le code suivant présente le modèle d'écouteur d'événement :

```
var listenerObject:Object = new Object();
listenerObject.eventName = function(eventObj:Object) {
  // Votre code ici.
};
broadcasterObject.addListener(listenerObject);
```

Le code commence par un objet, *`listenerObject`*, avec une propriété *`eventName`*. L'objet écouteur spécifié peut être un objet, tel qu'un objet existant, une occurrence de bouton ou de clip sur la scène ou une occurrence de n'importe quelle classe ActionScript. Par exemple, un clip personnalisé peut implémenter les méthodes d'écoute pour les écouteurs de la scène. Vous pouvez également avoir un objet qui écoute plusieurs types d'écouteurs.

La propriété *`eventName`* est un événement qui se produit sur *`broadcasterObject`*, qui ensuite diffuse l'événement à *`listenerObject`*. Vous pouvez enregistrer plusieurs écouteurs sur un diffuseur d'événement.

Affectez ensuite une fonction à l'écouteur d'événement qui répond en quelque sorte à l'événement.

Enfin, appelez la méthode `addListener()` sur l'objet qui diffuse l'événement, en lui transmettant le nom de l'objet écouteur.

Pour désenregistrer un objet écouteur de sorte qu'il ne reçoive plus d'événements, appelez la méthode `removeListener()` de l'objet diffuseur, en lui transmettant le nom de l'événement à supprimer et l'objet écouteur.

```
broadcasterObject.removeListener(listenerObject);
```

Exemple d'écouteur d'événement

L'exemple suivant illustre comment utiliser l'écouteur d'événement `onSetFocus` dans la classe Selection pour créer un gestionnaire de focus simple pour un groupe de champs de saisie de texte. Dans ce cas, la bordure du champ de texte qui reçoit le focus clavier est activée (affichée) et la bordure du champ de texte qui a perdu le focus est désactivée.

Pour créer un gestionnaire de focus simple avec des gestionnaires d'événement :

1. En utilisant l'outil Texte, créez un champ de texte sur la scène.
2. Sélectionnez le champ de texte et, dans l'inspecteur des propriétés, choisissez Entrée dans le menu contextuel Type de texte, puis cliquez sur le bouton Afficher la bordure autour du texte.
3. Créez un autre champ de saisie de texte, sous le premier.

   Assurez-vous que l'option Afficher la bordure autour du texte n'est pas activée pour ce champ de texte. Vous pouvez poursuivre la création de champs texte.
4. Sélectionnez l'image 1 dans le scénario et ouvrez le panneau Actions (Fenêtre > Actions).
5. Pour créer un objet qui écoute les notifications de focus d'une classe Selection, entrez le code suivant dans le panneau Actions :

```
// Crée un objet écouteur, focusListener.
var focusListener:Object = new Object();
// Définit la fonction pour l'objet écouteur.
focusListener.onSetFocus = function(oldFocus_txt:TextField,
  newFocus_txt:TextField) {
  oldFocus_txt.border = false;
  newFocus_txt.border = true;
}
```

   Ce code crée un objet nommé `focusListener` qui définit une propriété `onSetFocus` et lui affecte une fonction. La fonction tient compte de deux paramètres : une référence au champ de texte qui a perdu le focus et une référence au champ de texte qui a reçu le focus. La fonction définit la propriété `border` du champ de texte qui a perdu le focus sur `false` et la propriété `border` du champ de texte qui a reçu le focus sur `true`.

6. Pour enregistrer l'objet `focusListener` pour recevoir des événements de l'objet Selection, ajoutez le code suivant au panneau Actions :

```
// Enregistre focusListener auprès du diffuseur.
Selection.addListener(focusListener);
```

7. Testez l'application (Contrôle > Tester l'animation), cliquez dans le premier champ de texte, puis appuyez sur la touche de tabulation pour passer d'un champ à l'autre.

Utilisation d'écouteurs d'événement avec des composants

La syntaxe d'un écouteur d'événement change légèrement lorsque l'on utilise des composants. Les composants génèrent des événements, qui doivent être écoutés de façon spécifique par un objet écouteur ou par une fonction personnalisée.

L'exemple suivant présente l'utilisation d'écouteurs d'événement pour surveiller la progression du téléchargement dynamique d'une image.

Pour écouter les événements du composant Loader :

1. Faites glisser une occurrence du composant Loader du panneau Composants jusqu'à la scène.
2. Sélectionnez la nouvelle occurrence et saisissez **my_ldr** dans le champ Nom de l'occurrence de l'inspecteur des propriétés.
3. Ajoutez le code suivant à l'image 1 du scénario principal :

```
System.security.allowDomain("http://www.helpexamples.com");

var loaderListener:Object = new Object();
loaderListener.progress = function(evt_obj:Object):Void {
  trace(evt_obj.type); // progress
  trace("\t" + evt_obj.target.bytesLoaded + " of " +
  evt_obj.target.bytesTotal + " bytes loaded");
}
loaderListener.complete = function(evt_obj:Object):Void {
  trace(evt_obj.type); // complete
}

my_ldr.addEventListener("progress", loaderListener);
my_ldr.addEventListener("complete", loaderListener);
my_ldr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

Ce code ActionScript définit un objet écouteur nommé `loaderListener`, qui écoute deux événements : `progress` et `complete`. Lorsque chacun de ces événements est diffusé, son code est exécuté et le texte de débogage s'affiche dans le panneau de sortie si vous testez le fichier SWF dans l'outil de programmation.

Vous indiquez ensuite à l'occurrence `my_ldr` d'écouter chacun des deux événements spécifiés (`progress` et `complete`) et vous désignez l'objet écouteur ou la fonction à exécuter lorsque l'événement est diffusé. Enfin, la méthode `Loader.load()` est appelée, ce qui déclenche le téléchargement de l'image.

4. Choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

   L'image est téléchargée dans l'occurrence Loader de la scène, puis plusieurs messages s'affichent dans le panneau de sortie. Selon la taille de l'image téléchargée, et si l'image a été mise en cache sur l'ordinateur local de l'utilisateur, l'événement `progress` peut être diffusé plusieurs fois, alors que l'événement complete n'est diffusé qu'une fois à la fin du téléchargement de l'image.

   Lorsque vous utilisez des composants et diffusez des événements, la syntaxe diffère légèrement de celle des écouteurs d'événement des exemples précédents. En particulier, vous devez utiliser la méthode `addEventListener()` au lieu d'appeler `addListener()`. Deuxièmement, vous devez désigner l'événement spécifique que vous souhaitez écouter, ainsi que l'objet écouteur ou la fonction.

Au lieu d'utiliser un objet écouteur, comme à la section « Utilisation d'écouteurs d'événement avec des composants », à la page 319, vous pouvez également utiliser une fonction personnalisée. Le code de l'exemple précédent pourrait être réécrit de la manière suivante :

```
System.security.allowDomain("http://www.helpexamples.com");

my_ldr.addEventListener("progress", progressListener);
my_ldr.addEventListener("complete", completeListener);
my_ldr.load("http://www.helpexamples.com/flash/images/image1.png");

function progressListener(evt_obj:Object):Void {
  trace(evt_obj.type); // progress
  trace("\t" + evt_obj.target.bytesLoaded + " of " +
  evt_obj.target.bytesTotal + " bytes loaded");
}
function completeListener(evt_obj:Object):Void {
  trace(evt_obj.type); // complete
}
```

REMARQUE

Remarquez que, dans les exemples précédents, les écouteurs d'événement sont toujours ajoutés avant l'appel de la méthode `Loader.load()`. Si vous appelez la méthode `Loader.load()` avant de spécifier les écouteurs d'événement, il est possible que le téléchargement se termine avant que les écouteurs d'événement soient entièrement définis. Cela signifie que le contenu peut s'afficher et que l'événement `complete` peut ne pas être intercepté.

Association de gestionnaires d'événement à des boutons et des clips

Vous pouvez associer des gestionnaires d'événement directement à une occurrence de bouton ou de clip sur la scène au moyen des gestionnaires d'événement `onClipEvent()` et `on()`. Le gestionnaire d'événement `onClipEvent()` diffuse les événements de clip alors que le gestionnaire d'événement `on()` traite les événements de bouton.

Pour lier un gestionnaire d'événement à une occurrence de bouton ou de clip, cliquez sur cette occurrence sur la scène pour lui donner le focus, puis entrez du code dans le panneau Actions. Le titre du panneau Actions indique que du code va être lié au bouton ou au clip : panneau Actions - panneau Bouton ou Actions - Clip. Pour consulter les recommandations liées à l'exploitation de code avec des occurrences de bouton ou de clip, consultez la section « Association de code à des objets », à la page 732.

REMARQUE

Ne confondez pas les gestionnaires d'événement de boutons et de clips et les événements de composant, tels que `SimpleButton.click`, `UIObject.hide` et `UIObject reveal`, qui doivent être liés aux occurrences de composants et sont abordés dans le guide *Utilisation des composants ActionScript 2.0*.

Vous pouvez associer `onClipEvent()` et `on()` uniquement à des occurrences de clips qui ont été placées sur la scène au cours de la programmation. Il est impossible d'associer `onClipEvent()` ou `on()` à des occurrences de clips créées à l'exécution (avec la méthode `attachMovie`, par exemple). Pour associer des gestionnaires d'événement à des objets créés à l'exécution, utilisez des méthodes de gestionnaire d'événement ou des écouteurs d'événement. (Voir « ActionScript et les événements », à la page 312 et « Utilisation des écouteurs d'événement », à la page 316.)

REMARQUE

Il n'est pas conseillé de lier les gestionnaires d'événement `onClipEvent()` et `on()`. Il est préférable d'intégrer votre code à des scripts d'image ou à un fichier de classe, comme décrit tout au long de ce manuel. Pour plus d'informations, voir « ActionScript et les événements », à la page 312 et « Association de code à des objets », à la page 732.

Pour plus d'informations sur les gestionnaires d'événement de boutons et de clips, consultez les rubriques suivantes :

- « Utilisation de on et onClipEvent avec des méthodes de gestionnaires d'événement », à la page 322
- « Spécification d'événements pour les méthodes on ou onClipEvent », à la page 323
- « Ajout ou affectation de plusieurs gestionnaires à un même objet », à la page 324

Utilisation de on et onClipEvent avec des méthodes de gestionnaires d'événement

Vous pouvez, dans certains cas, appliquer différentes techniques pour gérer les événements sans provoquer de conflits. L'utilisation des méthodes `on()` et `onClipEvent()` ne provoque pas de conflit avec les méthodes de gestionnaire d'événement que vous définissez.

Imaginons par exemple que votre fichier SWF comporte un bouton associé à un gestionnaire `on(press)` qui déclenche la lecture du fichier SWF. Ce même bouton possède également une méthode `onPress()`, pour laquelle vous définissez une fonction qui fait pivoter un objet sur la scène. Lorsque l'utilisateur clique sur le bouton, la lecture du fichier SWF commence et l'objet tourne sur lui-même. Selon les types d'événement à appeler et le moment auquel ces derniers doivent être appelés, vous pouvez utiliser les méthodes `on()` et `onClipEvent()`, des méthodes de gestionnaire d'événement ou ces deux types de gestion d'événement.

Toutefois, le domaine des variables et des objets dans les gestionnaires `on()` et `onClipEvent()` n'est pas le même que celui du gestionnaire d'événement et des écouteurs d'événement. Voir « Domaine du gestionnaire d'événement », à la page 327.

Vous pouvez également utiliser `on()` avec des clips pour créer des clips qui reçoivent des événements de bouton. Pour plus d'informations, voir « Création de clips avec des états de bouton », à la page 326. Pour plus d'informations sur la spécification d'événements pour `on()` et `onClipEvent()`, consultez la section « Spécification d'événements pour les méthodes on ou onClipEvent », à la page 323.

Pour utiliser un gestionnaire on et le gestionnaire d'événements onPress :

1. Créez un nouveau document Flash, puis enregistrez-le sous le nom **handlers.fla**.
2. A l'aide de l'outil Rectangle, dessinez un grand carré sur la scène.
3. A l'aide de l'outil Sélection, double-cliquez sur le carré sur la scène, puis appuyez sur F8 pour ouvrir la boîte de dialogue Convertir en symbole.
4. Entrez un nom de symbole pour la zone, définissez le type clip et cliquez sur OK.
5. Donnez au clip sur la scène le nom d'occurrence **box_mc**.
6. Ajoutez le code ActionScript suivant directement au symbole du clip sur la scène :

```
on (press) {
  trace("on (press) {...}");
}
```

7. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
box_mc.onPress = function() {
  trace("box_mc.onPress = function() {...};");
};
```

8. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

   Lorsque vous cliquez sur le symbole du clip sur la scène, les informations suivantes s'affichent dans le panneau de sortie :

   on (press) {...}

   box_mc.onPress = function() {...};

REMARQUE

Il n'est pas conseillé de lier les gestionnaires d'événement `onClipEvent()` et `on()`. Il est préférable d'intégrer votre code à des scripts d'image ou à un fichier de classe, comme décrit tout au long de ce manuel. Pour plus d'informations, voir « ActionScript et les événements », à la page 312 et « Association de code à des objets », à la page 732.

Spécification d'événements pour les méthodes on ou onClipEvent

Pour utiliser le gestionnaire `on()` ou `onClipEvent()`, associez-le directement à une occurrence d'un bouton ou d'un clip sur la scène et spécifiez l'événement que vous souhaitez gérer pour cette occurrence. Pour obtenir la liste complète des événements pris en charge par les gestionnaires d'événement `on()` et `onClipEvent()`, consultez les entrées `gestionnaire on` et `gestionnaire onClipEvent` dans le *Guide de référence du langage ActionScript 2.0*.

Par exemple, le gestionnaire d'événement `on()` suivant est exécuté lorsque l'utilisateur clique sur le bouton auquel il est associé :

```
on (press) {
  trace("Thanks for pressing me.");
}
```

Vous pouvez spécifier plusieurs événements pour chaque gestionnaire `on()` en les séparant par des virgules. Le code ActionScript d'un gestionnaire est exécuté lorsque l'un des événements spécifiés par le gestionnaire se produit. Par exemple, le gestionnaire `on()` suivant, associé à un bouton, est exécuté lorsque le pointeur de la souris passe au-dessus du bouton, puis s'en écarte :

```
on (rollOver, rollOut) {
  trace("You rolled over, or rolled out");
}
```

Vous pouvez également ajouter des événements de touche enfoncée à l'aide de gestionnaires `on()`. Par exemple, le code suivant trace une chaîne lorsque vous appuyez sur la touche 3 du clavier. Sélectionnez une occurrence de bouton ou de clip, puis ajoutez le code suivant dans le panneau Actions :

```
on (keyPress "3") {
  trace("You pressed 3")
}
```

Ou, pour faire un suivi de la chaîne lorsque la touche Entrée est enfoncée par l'utilisateur, vous pouvez utiliser le format de code suivant : Sélectionnez une occurrence de bouton ou de clip, puis ajoutez le code suivant dans le panneau Actions :

```
on (keyPress "<Enter>") {
  trace("Enter Pressed");
}
```

Choisissez Contrôle > Tester l'animation, puis appuyez sur la touche Entrée pour visualiser le tracé de la chaîne dans le panneau de sortie. Si le tracé n'apparaît pas, sélectionnez Contrôle > Désactiver les raccourcis clavier, puis réessayez. Pour plus d'informations sur l'ajout d'une interactivité relative à la pression des touches à vos applications, consultez l'entrée `Key`.

REMARQUE

Il n'est pas conseillé de lier les gestionnaires d'événement `onClipEvent()` et `on()`. Il est préférable d'intégrer votre code à des scripts d'image ou à un fichier de classe, comme décrit tout au long de ce manuel. Pour plus d'informations, voir « ActionScript et les événements », à la page 312 et « Association de code à des objets », à la page 732.

Ajout ou affectation de plusieurs gestionnaires à un même objet

Vous pouvez associer plusieurs gestionnaires à un objet si vous souhaitez exécuter différents scripts lorsque différents événements se produisent. Par exemple, vous pouvez associer les gestionnaires `onClipEvent()` suivants à la même occurrence de clip. Le premier est exécuté au premier chargement du clip (ou lorsque celui-ci apparaît sur la scène), le second est exécuté lorsque le clip est purgé de la scène.

```
on (press) {
  this.unloadMovie()
}
onClipEvent (load) {
  trace("I've loaded");
}
onClipEvent (unload) {
  trace("I've unloaded");
}
```

REMARQUE

Il n'est pas conseillé de lier les gestionnaires d'événement `onClipEvent()` et `on()`. Il est préférable d'intégrer votre code à des scripts d'image ou à un fichier de classe, comme décrit tout au long de ce manuel. Pour plus d'informations, voir « ActionScript et les événements », à la page 312 et « Association de code à des objets », à la page 732.

Pour lier plusieurs gestionnaires à un objet à l'aide du code placé dans le scénario, reportez-vous à l'exemple suivant. Le code associe les gestionnaires `onPress` et `onRelease` à une occurrence de clip.

Pour affecter plusieurs gestionnaires à un même objet :

1. Créez un nouveau document Flash et nommez-le **assignMulti.fla**.
2. Sélectionnez l'image 1 du scénario, puis ajoutez le code suivant dans le panneau Actions :

```
this.createEmptyMovieClip("img_mc", 10);
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
     target_mc.onPress = function() {
          target_mc.startDrag();
     };
     target_mc.onRelease = function() {
          target_mc.stopDrag();
     };
}
mclListener.onLoadError = function(target_mc:MovieClip) {
     trace("error downloading image");
}
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);
```

3. Sélectionnez Contrôle > Tester l'animation pour tester le document.

   L'image se charge dans l'occurrence de `img_mc` et les gestionnaires d'événement `onPress()` et `onRelease()` vous permettent de faire glisser l'image sur la scène.

Diffusion d'événements à partir d'occurrences de composant

Quelle que soit l'occurrence de composant, vous pouvez spécifier le mode de gestion d'un événement. Les événements de composant sont traités différemment par rapport aux diffusions d'événements en provenance d'objets ActionScript natifs.

Pour plus d'informations, consultez *Utilisation des composants ActionScript 2.0*.

Création de clips avec des états de bouton

Lorsque vous associez un gestionnaire `on()` à un clip ou que vous affectez une fonction à l'un des gestionnaires d'événement souris MovieClip pour une occurrence de clip, le clip répond à des événements souris de la même façon qu'un bouton. Vous pouvez également créer des états de bouton automatiques (Relevé, Dessus et Abaissé) dans un clip, en ajoutant les étiquettes d'image `_up`, `_over` et `_down` au scénario du clip.

Lorsque l'utilisateur déplace la souris au-dessus d'un clip ou qu'il clique sur ce dernier, la tête de lecture passe à l'image comportant l'étiquette d'image appropriée. La propriété hitArea (propriété MovieClip.hitArea) permet de désigner la zone active d'un clip.

Pour créer des états de bouton dans un clip :

1. Créez un nouveau document Flash, puis enregistrez-le sous le nom **mcbutton.fla**.
2. A l'aide de l'outil Rectangle, tracez un petit rectangle (d'environ 100 pixels de large et 20 pixels de haut) sur la scène.
3. A l'aide de l'outil Sélection, double-cliquez sur la forme, puis appuyez sur F8 pour ouvrir la boîte de dialogue Convertir en symbole.
4. Entrez le nom de symbole **mcbutton**, définissez le type de symbole de clip et cliquez sur OK.
5. Double-cliquez sur le symbole de clip sur la scène pour activer le mode d'édition de symbole.
6. Créez un calque dans le scénario du clip, et nommez-le **labels**.
7. Entrez l'étiquette d'image `_up` dans l'inspecteur des propriétés.
8. Créez un nouveau calque sur le calque par défaut et le calque labels.
9. Renommez le nouveau calque **actions** et ajoutez le code ActionScript suivant à l'image 1 du scénario du clip :

   ```
   stop();
   ```

10. Sélectionnez l'image 10, les trois calques, puis sélectionnez Insertion > Scénario > Image-clé.
11. Ajoutez une action `stop()` à l'image 10 du calque actions, puis ajoutez l'étiquette d'image **_over** à l'image 10 du calque labels.
12. Sélectionnez le rectangle dans l'image 10 et sélectionnez une nouvelle couleur de remplissage à l'aide de l'inspecteur des propriétés.
13. Créez de nouvelles images-clés dans l'image 20 de chacun des trois calques, puis ajoutez l'étiquette d'image **_down** dans l'inspecteur des propriétés.

14. Modifiez la couleur du rectangle dans l'image 20 afin que chacun des trois états de bouton apparaisse dans une couleur différente.
15. Revenez au scénario principal.
16. Pour que votre clip réponde aux événements souris, effectuez l'une des opérations suivantes :
    - Ajoutez un gestionnaire d'événement `on()` à l'occurrence de clip, comme indiqué dans la section « Association de gestionnaires d'événement à des boutons et des clips », à la page 321.
    - Affectez une fonction à l'un des gestionnaires d'événement souris de l'objet du clip (`onPress`, `onRelease`, etc.), comme indiqué dans la section « ActionScript et les événements », à la page 312.
17. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

    Lorsque vous passez le pointeur de la souris sur une occurrence de clip sur la scène, le clip passe automatiquement à l'état _over. Cliquez sur l'occurrence de clip pour que la tête de lecture passe automatiquement à l'état _down du clip.

Domaine du gestionnaire d'événement

Le domaine, ou *contexte*, des variables et commandes que vous déclarez et exécutez dans un gestionnaire d'événement varie selon le type de gestionnaire d'événement que vous utilisez : gestionnaires d'événement ou écouteurs d'événement, gestionnaires `on()` et `onClipEvent()`. Si vous définissez un gestionnaire d'événement dans une nouvelle classe ActionScript 2.0, le domaine dépend également du mode de définition du gestionnaire d'événement. Cette section regroupe des exemples relatifs à ActionScript 1.0 et à ActionScript 2.0.

Exemples ActionScript 1.0 Les fonctions affectées aux méthodes de gestionnaire d'événement et aux écouteurs d'événement (comme toutes les fonctions ActionScript que vous rédigez) définissent un domaine de variable locale, contrairement aux gestionnaires `on()` et `onClipEvent()`.

Par exemple, soit les deux gestionnaires d'événement suivants : le premier est un gestionnaire d'événement `onPress` associé à un clip appelé `clip_mc`, le second est un gestionnaire `on()` associé à la même occurrence de clip.

```
// Ajouté au scénario de clip parent de clip_mc :
clip_mc.onPress = function () {
  var shoeColor; // Variable de la fonction locale
  shoeColor = "blue";
}
// Gestionnaire on() ajouté au clip_mc :
on (press) {
  var shoeColor; // aucun domaine de variable locale
  shoeColor = "blue";
}
```

Bien que les deux gestionnaires d'événement contiennent le même code, leurs résultats sont différents. Dans le premier cas, la variable `color` est locale par rapport à la fonction définie pour `onPress`. Dans le second, le gestionnaire `on()` ne définissant pas de domaine de variable locale, la variable est définie dans le domaine du scénario du clip `clip_mc`.

Dans le cas des gestionnaires d'événement `on()` associés à des boutons, plutôt qu'à des clips, le domaine des variables (à l'instar des appels de fonction et de méthode) est limité au scénario qui contient l'occurrence de bouton.

Le gestionnaire d'événement `on()` suivant, par exemple, génère différents résultats selon qu'il est ajouté à un clip ou à un bouton. Dans le premier exemple, l'appel de la fonction `play()` lance la tête de lecture du scénario qui contient le bouton ; dans le second, l'appel de la fonction `play()` démarre le scénario du clip auquel le gestionnaire est associé.

```
// Associé à un bouton.
on (press) {
  play(); // Exécute le scénario parent.
}
// Associé à un clip.
on (press) {
  play(); // Exécute le scénario du clip.
}
```

Autrement dit, lorsqu'elle est associée à un objet bouton, la fonction `play()` s'applique au scénario qui contient le bouton, c'est-à-dire au scénario parent du bouton. En revanche, lorsque le gestionnaire `on(press)` est associé à un objet de clip, la fonction `play()` s'applique au clip qui contient le gestionnaire. Si vous associez le code suivant à un clip, le scénario parent est joué :

```
// Associé à un clip.
on (press) {
  _parent.play(); // Exécute le scénario parent.
}
```

Dans la définition d'un gestionnaire d'événement ou d'écouteur d'événement, la fonction `play()` s'applique au scénario contenant la définition. Imaginons, par exemple, que vous déclariez la méthode de gestionnaire d'événement `my_mc.onPress` suivante dans le scénario contenant l'occurrence de clip `my_mc`.

```
// Fonction définie dans le scénario.
my_mc.onPress = function () {
  play(); // Lit le scénario ayant servi à la définition.
};
```

Pour exécuter le clip qui définit le gestionnaire d'événement `onPress`, vous devez faire explicitement référence à ce clip à l'aide du mot-clé `this`, comme suit :

```
// Fonction définie dans le scénario racine.
my_mc.onPress = function () {
  this.play(); // Joue le scénario de my_mc clip.
};
```

Toutefois, le même code placé sur le scénario racine d'une occurrence de bouton lirait le scénario racine :

```
my_btn.onPress = function () {
  this.play(); // Exécute le scénario racine
};
```

Pour plus d'informations sur le domaine du mot-clé `this` dans les gestionnaires d'événement, consultez la section « Domaine du mot-clé this », à la page 331.

Exemple ActionScript 2.0 La classe TextLoader suivante permet de charger un fichier texte et d'afficher un texte après avoir chargé le fichier.

```
// TextLoader.as
class TextLoader {
  private var params_lv:LoadVars;
  public function TextLoader() {
    params_lv = new LoadVars();
    params_lv.onLoad = onLoadVarsDone;
    params_lv.load("http://www.helpexamples.com/flash/params.txt");
  }
  private function onLoadVarsDone(success:Boolean):Void {
    _level0.createTextField("my_txt", 999, 0, 0, 100, 20);
    _level0.my_txt.autoSize = "left";
    _level0.my_txt.text = params_lv.monthNames; // non défini
  }
}
```

Ce code ne peut pas fonctionner correctement car il y a un problème de domaine avec les gestionnaires d'événement. Par ailleurs, la référence `this` prête à confusion entre le gestionnaire d'événement `onLoad` et la classe. Le comportement attendu dans cet exemple est que la méthode `onLoadVarsDone()` soit appelée dans le domaine de l'objet TextLoader, alors qu'elle est appelée dans le domaine de l'objet LoadVars. En effet, la méthode a été extraite à partir de l'objet TextLoader et greffée sur l'objet LoadVars. L'objet LoadVars appelle alors le gestionnaire d'événement `this.onLoad` lorsque le fichier texte a été chargé avec succès et lorsque la fonction `onLoadVarsDone()` appelée par `this` est définie sur LoadVars et non sur TextLoader. L'objet `params_lv` réside dans le domaine `this` lorsqu'il est appelé, bien que la fonction `onLoadVarsDone()` repose sur l'objet `params_lv` par référence. Par conséquent, la fonction `onLoadVarsDone()` attend une occurrence `params_lv.params_lv` qui n'existe pas.

Pour appeler correctement la méthode `onLoadVarsDone()` dans le domaine de l'objet TextLoader, appliquez la stratégie suivante : utilisez une fonction littérale pour créer une fonction anonyme qui appelle la fonction voulue. L'objet `owner` reste visible dans le domaine de la fonction anonyme, ce qui permet de rechercher l'objet TextLoader ayant procédé à l'appel.

```
// TextLoader.as
class TextLoader {
  private var params_lv:LoadVars;
  public function TextLoader() {
    params_lv = new LoadVars();
    var owner:TextLoader = this;
    params_lv.onLoad = function (success:Boolean):Void {
      owner.onLoadVarsDone(success);
    }
    params_lv.load("http://www.helpexamples.com/flash/params.txt");
  }
  private function onLoadVarsDone(success:Boolean):Void {
    _level0.createTextField("my_txt", 999, 0, 0, 100, 20);
    _level0.my_txt.autoSize = "left";
    _level0.my_txt.text = params_lv.monthNames; // Janvier, février,
  mars,...
  }
}
```

Domaine du mot-clé this

Le mot-clé `this` fait référence à l'objet dans le domaine en cours d'exécution. Selon le type de technique de gestionnaire d'événement que vous utilisez, `this` renvoie à différents objets.

Dans une fonction de gestionnaire d'événement ou d'écouteur d'événement, `this` fait référence à l'objet qui définit la méthode du gestionnaire d'événement ou de l'écouteur d'événement. Dans le code suivant, par exemple, `this` renvoie à `my_mc` :

```
// Gestionnaire d'événement onPress() associé au scénario principal :
my_mc.onPress = function () {
  trace(this); // _level0.my_mc
}
```

Dans un gestionnaire on() associé à un clip, `this` renvoie au clip auquel le gestionnaire `on()` est associé, comme indiqué dans le code suivant :

```
// Associé à un clip appelé my_mc sur le scénario principal.
on (press) {
  trace(this); // _level0.my_mc
}
```

Dans un gestionnaire on() associé à un bouton, `this` renvoie au scénario qui contient le bouton, comme dans le code suivant :

```
// Associé à un bouton du scénario principal.
on (press) {
  trace(this); // _level0
}
```

Utilisation de la classe 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 (consultez le guide *Utilisation des composants ActionScript 2.0*) et appeler des fonctions au sein du domaine de la classe qui les contient.

Lorsque vous transmettez une fonction sous forme de paramètre à la méthode `EventDispatcher.addEventListener()`, la fonction est invoquée dans le domaine de l'occurrence de composant diffuseur, pas de l'objet dans lequel elle est déclarée (consultez le guide *Utilisation des composants ActionScript 2.0*). Pour appeler la fonction dans le domaine de l'objet dans lequel elle est déclarée, vous pouvez utiliser `Delegate.create()`.

L'exemple suivant présente trois méthodes d'écoute d'événements pour une occurrence de composant Button. Chaque méthode d'ajout d'écouteurs d'événement à une occurrence de composant Button distribue l'événement vers un domaine différent.

Pour utiliser la classe Delegate pour écouter des événements :

1. Créez un document Flash, puis enregistrez-le sous le nom **delegate.fla**.
2. Faites glisser un composant Button du dossier Interface utilisateur du panneau Composants à la bibliothèque.

   Vous pourrez ajouter et positionner l'occurrence de bouton sur la scène ultérieurement à l'aide du code ActionScript.
3. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
import mx.controls.Button;
import mx.utils.Delegate;

function clickHandler(eventObj:Object):Void {
  trace("[" + eventObj.type + "] event on " + eventObj.target + "
  instance.");
  trace("\t this -> " + this);
}

var buttonListener:Object = new Object();
buttonListener.click = function(eventObj:Object):Void {
  trace("[" + eventObj.type + "] event on " + eventObj.target + "
  instance.");
  trace("\t this -> " + this);
};

this.createClassObject(Button, "one_button", 10, {label:"One"});
one_button.move(10, 10);
one_button.addEventListener("click", clickHandler);

this.createClassObject(Button, "two_button", 20, {label:"Two"});
two_button.move(120, 10);
two_button.addEventListener("click", buttonListener);

this.createClassObject(Button, "three_button", 30, {label:"Three"});
three_button.move(230, 10);
three_button.addEventListener("click", Delegate.create(this,
  clickHandler));
```

Le code qui précède est divisé en six sections (séparées par une ligne vide). La première section importe la classe Button (pour le composant Button) ainsi que la classe Delegate. La deuxième section du code définit la fonction qui est appelée lorsque l'utilisateur clique sur certains boutons. La troisième section de code crée un objet que vous utilisez comme un écouteur d'événement ; il écoute un seul événement, `click`.

Les trois dernières sections de code créent chacune une nouvelle occurrence du composant Button sur la scène, repositionnent l'occurrence et ajoutent un écouteur d'événement pour l'événement `click`. Le premier bouton ajoute un écouteur pour l'événement `click` et transmet directement une référence à une fonction de gestionnaire `click`. Le deuxième bouton ajoute un écouteur pour l'événement `click` et transmet une référence à un objet écouteur qui contient un gestionnaire pour l'événement `click`. Enfin, la troisième fonction ajoute un écouteur pour l'événement `click`, utilise la classe Delegate pour distribuer l'événement click dans le domaine `this` (où `this` équivaut à `_level0`) et transmet une référence à la fonction de gestionnaire `click`.

4. Choisissez Contrôle > Tester l'animation pour tester le document Flash.
5. Cliquez sur chaque occurrence de bouton sur la scène pour visualiser le domaine dans lequel l'événement est pris en charge.
   a. Cliquez sur le premier bouton sur la scène pour suivre le texte suivant dans le panneau de sortie :

      ```
      [click] event on _level0.one_button instance.
         this -> _level0.one_button
      ```

      Lorsque vous cliquez sur l'occurrence `one_button`, le domaine `this` fait directement référence à l'occurrence du bouton.

   b. Cliquez sur le deuxième bouton sur la scène pour suivre le texte suivant dans le panneau de sortie :

      ```
      [click] event on _level0.two_button instance.
         this -> [object Object]
      ```

      Lorsque vous cliquez sur l'occurrence `two_button`, le domaine `this` fait référence à l'objet `buttonListener`.

   c. Cliquez sur le troisième bouton sur la scène pour suivre le texte suivant dans le panneau de sortie :

      ```
      [click] event on _level0.three_button instance.
         this -> _level0
      ```

      Lorsque vous cliquez sur l'occurrence `three_button`, le domaine `this` fait référence au domaine spécifié dans l'appel de la méthode `Delegate.create()` soit, dans ce cas, `_level0`.

CHAPITRE 10

Utilisation des clips

10

Les clips sont comme des fichiers SWF autonomes exécutés indépendamment les uns des autres et du scénario qui les contient. Par exemple, si le scénario principal contient une seule image et qu'un clip contenu dans cette image comporte dix images, chaque image du clip est lue lorsque vous lisez le fichier SWF principal. Un clip peut lui-même contenir d'autres clips, ou *clips imbriqués*. Les clips imbriqués de cette manière sont organisés hiérarchiquement : le *clip parent* contient un ou plusieurs *clips enfants*.

Vous pouvez nommer des occurrences de clip pour les identifier en tant qu'objets contrôlables avec du code ActionScript. Lorsque vous donnez un *nom d'occurrence* à une occurrence de clip, ce nom l'identifie en tant qu'objet du type de classe MovieClip. Utilisez les propriétés et méthodes de la classe MovieClip pour contrôler l'apparence et le comportement des clips à l'exécution.

Vous pouvez considérer les clips comme des objets autonomes qui répondent à des événements, envoient des messages à d'autres objets de clip, conservent leur état et gèrent leurs clips enfants. De cette amnière, les clips constituent la base de l'*architecture basée sur les composants* de Flash CS3 Professional. Les composants disponibles dans le panneau Composants (Fenêtre > Composants) sont en réalité des clips sophistiqués qui ont été conçus et programmés pour adopter une apparence donnée et certains comportements.

Pour plus d'informations sur l'utilisation de l'API de dessin (méthodes de dessin de la classe MovieClip), les filtres, les mélanges, les animations scriptées, etc., consultez le Chapitre 12, « Animation, Filtres et Dessins ».

Pour plus d'informations sur les clips, consultez les rubriques suivantes :

A propos du contrôle des clips à l'aide d'ActionScript

Vous pouvez utiliser les fonctions ActionScript globales ou les méthodes de la classe MovieClip pour accomplir des tâches sur les clips. Certaines méthodes de la classe MovieClip accomplissent les mêmes tâches que les fonctions du même nom, alors que d'autres méthodes MovieClip, comme `hitTest()` et `swapDepths()`, ne possèdent pas de noms de fonction correspondants.

L'exemple suivant illustre la différence entre l'emploi d'une méthode et d'une fonction. Les deux instructions copient l'occurrence `my_mc`, nomment le nouveau clip `new_mc` et le placent à une profondeur de 5.

```
my_mc.duplicateMovieClip("new_mc", 5);
duplicateMovieClip(my_mc, "new_mc", 5);
```

Lorsqu'une fonction et une méthode présentent des comportements analogues, vous pouvez choisir l'une ou l'autre pour contrôler des clips. Le choix dépend de vos préférences et de votre familiarité avec la rédaction de scripts dans ActionScript. Que vous utilisiez une fonction ou une méthode, le scénario cible doit être chargé dans Flash Player lorsque la fonction ou la méthode est appelée.

Pour utiliser une méthode, vous l'invoquez en utilisant le chemin cible du nom d'occurrence, suivi d'un point (.), puis du nom et des paramètres de la méthode, comme dans l'exemple suivant :

```
myMovieClip.play();
parentClip.childClip.gotoAndPlay(3);
```

Dans la première instruction, la méthode `play()` lance la tête de lecture dans l'occurrence `myMovieClip`. Dans la deuxième instruction, la méthode `gotoAndPlay()` place la tête de lecture sur `childClip` (qui est un enfant de l'occurrence `parentClip`) à l'image 3 et continue la lecture.

Les fonctions globales qui contrôlent un scénario possèdent un paramètre *`cible`* qui permet de définir le chemin cible de l'occurrence que vous voulez contrôler. Par exemple, dans le script suivant, `startDrag()` cible l'occurrence recevant le code et la rend déplaçable :

```
my_mc.onPress = function() {
  startDrag(this);
};
my_mc.onRelease = function() {
  stopDrag();
};
```

Les fonctions suivantes ciblent des clips : `loadMovie()`, `unloadMovie()`, `loadVariables()`, `setProperty()`, `startDrag()`, `duplicateMovieClip()` et `removeMovieClip()`. Pour utiliser ces fonctions, vous devez entrer un chemin cible dans leur paramètre *`target`* pour indiquer la cible de la fonction.

Les méthodes MovieClip suivantes peuvent contrôler des clips ou des niveaux chargés et n'ont pas de fonctions équivalentes : `Clip.attacherFilm()`, `Clip.créerClipVide()`, `Clip.créerChampTexte()`, `MovieClip.getBounds()`, `MovieClip.getBytesLoaded()`, `MovieClip.getBytesTotal()`, `MovieClip.getDepth()`, `Clip.ExempleProfondeur()`, `Clip.ProfondeurMax.Suivante()`, `Clip.globalVersLocal()`, `Clip.localVersGlobal()`, `Clip.Test()`, `Clip.réglerMasque()`, `Clip.changerProfondeurs()`.

Pour plus d'informations sur ces fonctions et ces méthodes, consultez les entrées correspondantes dans le *Guide de référence du langage ActionScript 2.0*.

Pour un exemple de fichier source animation.fla qui décrit la manipulation des tableaux à l'aide du code ActionScript, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Animation afin d'accéder à l'exemple.

Pour des exemples d'applications de galerie de photos, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Galleries afin d'accéder à ces exemples.

- gallery_tree.fla
- gallery_tween.fla

Ces fichiers vous montrent comment utiliser ActionScript pour contrôler dynamiquement des clips tout en chargeant des fichiers image dans un fichier SWF avec des animations scriptées.

Appel de plusieurs méthodes sur un seul clip

Vous pouvez utiliser l'instruction `with` pour appeler un clip une seule fois, puis exécuter une série de méthodes sur ce clip. L'instruction `with` fonctionne sur tous les objets ActionScript (tels que Array, Color et Sound) et pas uniquement sur les clips.

L'instruction `with` prend un clip comme paramètre. L'objet que vous spécifiez est ajouté à la fin du chemin cible courant. Toutes les actions imbriquées dans une instruction `with` sont exécutées à l'intérieur du nouveau chemin cible. Par exemple, dans le script suivant, l'objet `donut.hole` passe au `avec l'instruction` pour changer les propriétés de `hole` :

```
with (donut.hole) {
  _alpha = 20;
  _xscale = 150;
  _yscale = 150;
}
```

Le script se comporte comme si les instructions imbriquées dans l'instruction `with` étaient appelées depuis le scénario de l'occurrence `hole`. Le code ci-dessus est équivalent à l'exemple suivant :

```
donut.hole._alpha = 20;
donut.hole._xscale = 150;
donut.hole._yscale = 150;
```

Le code ci-dessus est également équivalent à l'exemple suivant :

```
with (donut) {
  hole._alpha = 20;
  hole._xscale = 150;
  hole._yscale = 150;
}
```

Chargement et déchargement des fichiers SWF

Pour lire des fichiers SWF supplémentaires sans fermer Flash Player, ou pour faire permuter des fichiers SWF sans charger une autre page HTML, utilisez l'une des options suivantes :

- la fonction globale `loadMovie()` ou la méthode `loadMovie()` de la classe MovieClip.
- la méthode de la classe `loadClip()` MovieClipLoader. Pour plus d'informations sur la classe MovieClipLoader, consultez `MovieClipLoader` dans le *Guide de référence du langage ActionScript 2.0*.

Vous pouvez également utiliser la méthode `loadMovie()` pour envoyer des variables à un script CGI, ce qui génère un fichier SWF en tant que sortie CGI. Par exemple, elle permet de charger des fichiers dynamiques SWF ou des images en fonction des variables spécifiées dans un clip. Lorsque vous chargez un fichier SWF, vous pouvez spécifier comme cible un niveau ou un clip, dans lequel il sera chargé. Si vous chargez un fichier SWF dans une cible, il hérite des propriétés du clip ciblé. Une fois l'animation chargée, vous pouvez modifier ces propriétés.

La méthode `unloadMovie()` supprime un fichier SWF précédemment chargé avec la méthode `loadMovie()`. En purgeant explicitement les fichiers SWF avec `unloadMovie()`, vous assurez une transition fluide entre les fichiers SWF et vous allégez la quantité de mémoire requise par Flash Player. Il peut être plus efficace dans certains cas de définir la propriété `_visible` du clip sur `false` au lieu de le décharger. Si vous avez à réutiliser le clip par la suite, définissez la propriété `_visible` sur `false`, puis sur `true` en fonction des besoins.

Utilisez `loadMovie()` pour effectuer les opérations suivantes :

- Jouez une séquence de bannières publicitaires, correspondant à des fichiers SWF, en plaçant une fonction `loadMovie()` dans un fichier SWF conteneur qui charge et décharge de façon séquentielle les fichiers SWF de bannières.
- Développez une interface arborescente avec des liens qui permettent à l'utilisateur de sélectionner plusieurs fichiers SWF servant à afficher le contenu d'un site.
- Créez une interface de navigation, avec des contrôles de navigation au niveau 0, qui charge du contenu dans d'autres niveaux. Le chargement de contenu dans des niveaux permet de bénéficier de transitions plus souples entre les pages de contenu que le chargement de nouvelles pages HTML dans un navigateur.

Pour plus d'informations sur le chargement de fichiers SWF, consultez la section « Chargement de fichiers SWF et de fichiers d'images externes », à la page 591.

Pour plus d'informations, se reporter aux sections suivantes :

- « Spécification d'un scénario racine pour les fichiers SWF chargés », à la page 340
- « Chargement de fichiers image dans des clips », à la page 341

Spécification d'un scénario racine pour les fichiers SWF chargés

La propriété ActionScript `_root` spécifie ou contient une référence au scénario racine d'un fichier SWF. Si un fichier SWF possède plusieurs niveaux, le scénario racine se situe dans le niveau contenant le script en cours d'exécution. Par exemple, si un script de niveau 1 est évalué comme `_root`, `_level1` est renvoyé. Cependant, le scénario spécifié par `_root` peut changer si le fichier SWF est exécuté de façon indépendante (à son propre niveau) ou s'il a été chargé dans une occurrence de clip par un appel `loadMovie()`.

Dans l'exemple suivant, prenez un fichier nommé container.swf qui possède une occurrence de clip nommée `target_mc` dans son scénario principal. Le fichier container.swf déclare une variable nommée `userName` dans son scénario principal ; le même script charge ensuite un autre fichier nommé contents.swf dans l'occurrence de clip `target_mc` :

```
// Dans container.swf:
_root.userName = "Tim";
target_mc.loadMovie("contents.swf");
my_btn.onRelease = function():Void {
  trace(_root.userName);
};
```

Dans l'exemple suivant, le fichier SWF chargé contents.swf déclare également une variable nommée `userName` dans son scénario racine :

```
// Dans contents.swf :
_root.userName = "Mary";
```

Lorsque contents.swf est chargé dans le clip de container.swf, la valeur de `userName` associée au scénario racine du fichier SWF hébergeur (container.swf) prend la valeur « `Mary` » qui remplace « `Tim` ». Ceci peut entraîner le mauvais fonctionnement du code dans container.swf (ainsi que dans contents.swf).

Pour obliger `_root` à évaluer systématiquement le scénario du fichier SWF chargé, et non le scénario racine réel, utilisez la propriété `_lockroot`. Vous pouvez définir cette propriété depuis le fichier SWF en cours de chargement ou dans le fichier SWF qui lance le chargement. Lorsque `_lockroot` est défini sur `true` pour une occurrence de clip, le clip agira comme `_root` pour tout fichier SWF qui y sera chargé. Lorsque `_lockroot` est défini sur `true` au sein d'un fichier SWF, le fichier SWF en question agira comme sa propre racine, quel que soit l'autre fichier SWF effectuant le chargement. N'importe quel clip et n'importe quel nombre de clips peuvent définir `_lockroot` sur `true`. Cette propriété est `false` par défaut.

Par exemple, le créateur de container.swf peut entrer le code suivant sur l'image 1 du scénario principal :

```
// Ajouté à l'image 1 dans container.swf :
target_mc._lockroot = true;
```

Cette étape permet de s'assurer que toute référence à `_root` dans contents.swf, ou tout autre fichier SWF chargé dans `target_mc`, se rapporte à son propre scénario et non pas au scénario racine de container.swf. Désormais, « `Tim` » s'affiche lorsque vous cliquez sur le bouton.

L'auteur de contents.swf peut également ajouter le code suivant à son scénario principal :

```
// Ajouté à l'image 1 dans contents.swf :
this._lockroot = true;
```

Ceci garantit que, quel que soit l'endroit où contents.swf est chargé, toute référence à `_root` fera référence à son propre scénario principal et non à celui du fichier SWF hébergeur.

Pour plus d'informations, consultez _lockroot (propriété MovieClip._lockroot).

Chargement de fichiers image dans des clips

Vous pouvez utiliser la fonction `loadMovie()` ou la méthode MovieClip sur un même nom pour charger des fichiers image dans une occurrence de clip. Vous pouvez également utiliser la fonction `loadMovieNum()` pour charger un fichier image dans un niveau.

Lorsque vous chargez une image dans un clip, son coin supérieur gauche est placé au niveau du point de réglage du clip. Ce point se trouvant souvent au centre du clip, il se peut que l'image chargée ne soit pas centrée. De même, lorsque vous chargez une image dans un scénario principal, son coin supérieur gauche est placé dans le coin supérieur gauche de la scène. L'image chargée hérite des paramètres de rotation et de l'échelle du clip, mais le contenu initial du clip est supprimé.

Pour plus d'informations, consultez les entrées `fonction loadMovie`, loadMovie(`méthode MovieClip;.loadMovie)` et `fonction loadMovieNum` dans le *Guide de référence du langage ActionScript 2.0* et la section « Chargement de fichiers SWF et de fichiers d'images externes », à la page 591.

Modification de la position et de l'apparence d'un clip

Pour modifier les propriétés d'un clip pendant sa lecture, vous pouvez rédiger une instruction affectant une valeur à une propriété ou utiliser la fonction `setProperty()`. Par exemple, le code suivant fixe la rotation de l'occurrence `my_mc` à 45 :

```
my_mc._rotation = 45;
```

Cela équivaut au code suivant, qui utilise la fonction `setProperty()` :

```
setProperty("my_mc", _rotation, 45);
```

Certaines propriétés, appelées *propriétés en lecture seule*, ont des valeurs que vous pouvez lire mais pas définir. (Ces propriétés sont identifiées comme telles dans le guide *Guide de référence du langage ActionScript 2.0*). Les propriétés suivantes sont en lecture seule : `_currentframe`, `_droptarget`, `_framesloaded`, `_parent`, `_target`, `_totalframes`, `_url`, `_xmouse` et `_ymouse`.

Vous pouvez rédiger des instructions pour définir des propriétés qui ne sont pas en lecture seule. L'instruction suivante définit la propriété `_alpha` de l'occurrence de clip `wheel_mc`, qui est un enfant de l'occurrence `car_mc` :

```
car_mc.wheel_mc._alpha = 50;
```

En outre, vous pouvez rédiger des instructions qui récupèrent la valeur d'une propriété de clip. Par exemple, l'instruction suivante récupère la valeur de la propriété `_xmouse` dans le scénario du niveau actuel et définit la propriété `_x` de l'occurrence `my_mc`à cette valeur :

```
this.onEnterFrame = function() {
  my_mc._x = _root._xmouse;
};
```

Cela équivaut au code suivant, qui utilise la fonction `getProperty()`:

```
this.onEnterFrame = function() {
  my_mc._x = getProperty(_root, _xmouse);
};
```

Les propriétés `_x`, `_y`, `_rotation`, `_xscale`, `_yscale`, `_height`, `_width`, `_alpha` et `_visible` sont affectées par les transformations effectuées sur le parent du clip et transforment le clip et tous ses enfants. Les propriétés `_focusrect`, `_highquality`, `_quality` et `_soundbuftime` sont globales ; elles appartiennent uniquement au scénario principal de niveau 0. Toutes les autres propriétés appartiennent à chaque clip ou niveau chargé.

Pour consulter la liste des propriétés de clip, consultez le récapitulatif de propriété pour la classe MovieClip dans la section *Guide de référence du langage ActionScript 2.0*.

Pour un exemple de fichier source animation.fla qui décrit la manipulation des tableaux à l'aide du code ActionScript, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Animation afin d'accéder à l'exemple.

Pour des exemples d'applications de galeries de photos, consultez la page des exemples Flash à l'adresse twww.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Galleries afin d'accéder à ces exemples.

- gallery_tree.fla
- gallery_tween.fla

Déplacement des clips

Vous pouvez utiliser la fonction globale `startDrag()` ou la méthode `MovieClip.startDrag()` pour rendre un clip déplaçable. Par exemple, vous pouvez créer un clip pouvant être déplacé pour les jeux, les fonctions glisser-déposer, les interfaces personnalisables, les barres de défilement et les curseurs de défilement.

Un clip reste déplaçable jusqu'à son arrêt explicite par `stopDrag()` ou jusqu'à ce qu'un autre clip soit ciblé avec `startDrag()`. Vous ne pouvez déplacer qu'un seul clip à la fois dans un fichier SWF.

Pour créer des comportements plus complexes avec les opérations glisser-déposer, vous pouvez évaluer la propriété `_droptarget` du clip en cours de déplacement. Par exemple, vous pouvez examiner la propriété `_droptarget` pour voir si le clip a été déplacé vers un clip spécifique (tel qu'un clip « poubelle »), puis déclencher une autre action, comme dans l'exemple suivant :

```
// Faire glisser un élément vers la corbeille
garbage_mc.onPress = function() {
  this.startDrag(false);
};
// Lorsque cet élément arrive sur la corbeille, le rendre invisible
garbage_mc.onRelease = function() {
  this.stopDrag();
  // Convertit la notation à barre oblique en notation à point avec eval
  if (eval(this._droptarget) == trashcan_mc) {
    garbage_mc._visible = false;
  }
};
```

Pour plus d'informations, consultez les entrées `fonction startDrag` ou startDrag(`méthode MovieClip.startDrag` dans le *Guide de référence du langage ActionScript 2.0*.

Pour un exemple de fichier source, gallery_tween.fla, qui décrit l'utilisation du code ActionScript pour contrôler les clips dynamiquement au cours du chargement de fichiers images dans un fichier SWF et qui inclut la possibilité de rendre chaque clip déplaçable, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Galleries afin d'accéder à l'exemple.

Création de clips à l'exécution

En plus de créer des occurrences de clip dans l'environnement de programmation Flash, vous pouvez également créer des occurrences de clip lors de l'exécution de l'une des façons suivantes :

- « Création d'un clip vide », à la page 344
- « Duplication ou suppression d'un clip », à la page 346
- « Association d'un symbole de clip à la scène », à la page 346

Chaque occurrence de clip créée à l'exécution doit avoir un nom d'occurrence et une valeur de profondeur (ordre d'empilement ou ordre *z*). La profondeur que vous spécifiez détermine la façon dont le nouveau clip recouvre les autres clips sur le même scénario. Elle permet également d'écraser les clips qui résident à la même profondeur. (Voir la section « Gestion des profondeurs de clip », à la page 350.)

Pour un exemple de fichier source, gallery_tween.fla, qui décrit l'utilisation du code ActionScript pour contrôler les clips dynamiquement au cours du chargement de fichiers images dans un fichier SWF, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Galleries afin d'accéder à l'exemple.

Pour un exemple de fichier source, animation.fla, qui crée et supprime de nombreux clips à l'exécution, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Animation afin d'accéder à l'exemple.

Pour plus d'informations, voir les sections suivantes :

- « Création d'un clip vide », à la page 344
- « Duplication ou suppression d'un clip », à la page 346
- « Association d'un symbole de clip à la scène », à la page 346

Création d'un clip vide

Pour créer une occurrence de clip vide sur la scène, utilisez la méthode `createEmptyMovieClip()` de la classe MovieClip. Cette méthode crée un clip en tant qu'enfant du clip qui l'a appelée. Le point d'enregistrement d'un clip vide nouvellement créé se situe dans le coin supérieur gauche.

Par exemple, le code suivant permet de créer un nouveau clip enfant nommé `new_mc` à une profondeur de 10 dans le clip nommé `parent_mc` :

```
parent_mc.createEmptyMovieClip("new_mc", 10);
```

Le code suivant permet de créer un clip nommé `canvas_mc` sur le scénario racine du fichier SWF dans lequel le script est exécuté, puis d'activer `loadMovie()` pour y charger un fichier JPEG externe

```
this.createEmptyMovieClip("canvas_mc", 10);
canvas_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
```

Comme le montre l'exemple suivant, vous pouvez charger l'image image2.jpg dans un clip et utiliser la méthode `MovieClip.onPress()` pour que l'image se comporte comme un bouton. Le chargement d'une image avec `loadMovie()` remplace le clip par l'image, mais ne permet pas d'accéder aux méthodes du clip. Pour accéder aux méthodes du clip, vous devez créer un clip parent vide et un clip conteneur enfant. Chargez l'image dans ce conteneur et placez le gestionnaire d'événement dans le clip parent.

```
// Crée un clip parent pour stocker le conteneur.
this.createEmptyMovieClip("my_mc", 0);

// Crée un clip enfant dans « my_mc ».
// Clip devant être remplacé par l'image.
my_mc.createEmptyMovieClip("container_mc",99);

// Utiliser MovieClipLoader pour charger l'image.
var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image2.jpg",
  my_mc.container_mc);

// Placer le gestionnaire d'événement sur le clip parent my_mc.
my_mc.onPress = function():Void {
  trace("It works");
};
```

Pour plus d'informations, consultez l'entrée createEmptyMovieClip (`méthode MovieClip.createEmptyMovieClip` dans le *Guide de référence du langage ActionScript 2.0*.

Pour un exemple de fichier source, animation.fla, qui crée et supprime de nombreux clips à l'exécution, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Animation afin d'accéder à l'exemple.

Duplication ou suppression d'un clip

Pour dupliquer ou supprimer des occurrences de clip, utilisez les fonctions globales `duplicateMovieClip()` ou `removeMovieClip()`, ou encore les méthodes de la classe MovieClip du même nom. La méthode `duplicateMovieClip()` crée une nouvelle occurrence d'une occurrence de clip existante, lui affecte un nouveau nom d'occurrence et lui donne une profondeur, ou ordre *z*. Un clip dupliqué commence toujours à l'image 1 même si le clip initial se trouvait dans une autre image lors de la duplication, et il se situe toujours devant tous les clips prédéfinis placés dans le scénario.

Pour supprimer un clip que vous avez créé avec `duplicateMovieClip()`, utilisez `removeMovieClip()`. Un clip dupliqué est également supprimé si le clip parent est supprimé.

Pour plus d'informations, consultez les entrées `fonction duplicateMovieClip` et `fonction removeMovieClip` dans le *Guide de référence du langage ActionScript 2.0*.

Pour un exemple de fichier source, animation.fla, qui crée et supprime de nombreux clips à l'exécution, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Animation afin d'accéder à l'exemple.

Association d'un symbole de clip à la scène

La dernière manière de créer des occurrences de clip à l'exécution est d'utiliser la méthode `attachMovie()`. La méthode `attachMovie()` associe à la scène une occurrence de symbole de clip incluse dans la bibliothèque du fichier SWF. Le nouveau clip devient un clip enfant du clip l'ayant associé.

Pour associer un symbole de clip à partir de la bibliothèque avec du code ActionScript, vous devez exporter le symbole pour ActionScript et lui affecter un identifiant de liaison unique. Pour ce faire, utilisez la boîte de dialogue Propriétés de liaison.

Par défaut, tous les clips exportés pour être utilisés avec ActionScript sont chargés avant la première image du fichier SWF les contenant. Ce chargement peut entraîner un retard avant la lecture de la première image. Lorsque vous affectez un identifiant de liaison à un élément, vous pouvez également spécifier si cet élément doit être ajouté avant la première image. S'il n'est pas ajouté à la première image, vous devez en inclure une occurrence sur une autre image du fichier SWF ; sinon, l'élément ne sera pas exporté dans le fichier SWF.

Pour affecter un identifiant de liaison à un clip :

1. Choisissez Fenêtre > Bibliothèque pour ouvrir le panneau Bibliothèque.
2. Sélectionnez un clip dans le panneau Bibliothèque.
3. Dans le panneau Bibliothèque, choisissez Liaison dans le menu contextuel.

   La boîte de dialogue Propriétés de liaison apparaît.
4. Pour Liaison, activez l'option Exporter pour ActionScript.
5. Pour Identifiant, entrez l'identifiant du clip.

   Par défaut, l'identifiant est identique au nom du symbole.

   Vous pouvez également affecter une classe ActionScript au symbole de clip. Ceci permet au clip d'hériter des méthodes et des propriétés de la classe spécifiée. (Voir la section « Affectation d'une classe à un symbole de clip », à la page 363.)
6. Si vous ne souhaitez pas que le clip soit chargé avant la première image, désactivez l'option Exporter dans la première image.

   Si vous désactivez cette option, placez une occurrence du clip sur l'image du scénario où vous souhaitez qu'elle soit disponible. Par exemple, si le script que vous écrivez ne fait pas référence au clip avant l'image 10, placez une occurrence du symbole au niveau de l'image 10 ou juste avant celle-ci dans le scénario.
7. Cliquez sur OK.

Une fois que vous avez affecté un identifiant de liaison à un clip, vous pouvez associer une occurrence du symbole à la scène lors de l'exécution à l'aide de la méthode `attachMovie()`.

Pour associer un clip à un autre clip :

1. Affectez un identifiant de liaison à un symbole de bibliothèque de clip comme décrit dans l'exemple précédent.
2. Le panneau Actions étant ouvert (Fenêtre > Actions), sélectionnez une image dans le scénario.
3. Dans la fenêtre de script du panneau Actions, tapez le nom du clip ou du niveau auquel vous souhaitez associer le nouveau clip.

   Par exemple, pour associer le clip au scénario racine, tapez **ceci**.
4. Dans la boîte à outils Actions (à gauche du panneau Actions), cliquez sur les méthodes ActionScript 2.0 Classes > Movie > MovieClip > et sélectionnez `attachMovie()`.

5. A l'aide des conseils de code, entrez les valeurs des paramètres suivants :
   - Pour `idName`, spécifiez l'identifiant que vous avez saisi dans la boîte de dialogue Propriétés de liaison.
   - Pour `newName`, entrez un nom d'occurrence pour le clip associé afin de pouvoir le cibler.
   - Pour `depth`, entrez le niveau dans lequel l'animation dupliquée sera associée au clip. Chaque animation associée suit un ordre d'empilement qui lui est propre, le niveau 0 étant le niveau de l'animation d'origine. Les clips associés sont toujours au-dessus du clip d'origine, comme indiqué dans l'exemple suivant :
     ```
     this.attachMovie("calif_id", "california_mc", 10);
     ```

Pour plus d'informations, consultez l'entrée attachMovie (`méthode MovieClip.attachMovie)` dans le*Guide de référence du langage ActionScript 2.0*.

Ajout de paramètres aux clips créés dynamiquement

Lorsque vous créez ou dupliquez dynamiquement un clip à l'aide de `MovieClip.attachMovie()` et `MovieClip.duplicateMovie()`, vous pouvez remplir le clip avec des paramètres provenant d'un autre objet. Le paramètre *`objetInit`* de `attachMovie()` et `duplicateMovie()` permet aux clips créés dynamiquement de recevoir des paramètres de clip.

Pour plus d'informations, consultez attachMovie (méthode MovieClip.attachMovie) et `duplicateMovieClip (méthode MovieClip.duplicateMovieClip)` dans le *Guide de référence du langage ActionScript 2.0*.

Pour remplir un clip créé de façon dynamique avec des paramètres provenant d'un objet spécifié :

Effectuez l'une des opérations suivantes :

- Utilisez la syntaxe suivante avec `attachMovie()` :
  ```
  myMovieClip.attachMovie(idName, newName, depth [, initObject]);
  ```
- Utilisez la syntaxe suivante avec `duplicateMovie()`:
  ```
  myMovieClip.duplicateMovie(idName, newName, depth [, initObject]);
  ```

Le paramètre *`initObject`* spécifie le nom de l'objet dont vous souhaitez utiliser les paramètres pour remplir le clip créé dynamiquement.

Pour remplir un clip avec des paramètres en utilisant attachMovie() :

1. Dans un nouveau document Flash, créez un symbole de clip en choisissant Insertion > Nouveau symbole.
2. Tapez **dynamic_mc** dans la zone de texte Nom du symbole et sélectionnez le comportement de clip.
3. A l'intérieur du symbole, créez un champ de texte dynamique sur la scène avec **nom_txt** comme nom d'occurrence .

   Assurez-vous que ce champ texte est en dessous et à droite du point d'alignement.
4. Sélectionnez la première image du scénario du clip et ouvrez le panneau Actions (Fenêtre > Actions).
5. Créez une nouvelle variable nommée `nom_str` et affectez sa valeur à la propriété du `texte` de `nom_txt`, comme dans l'exemple suivant :

   ```
   var name_str:String;
   name_txt.text = name_str;
   ```
6. Choisissez Edition > Modifier le document pour revenir au scénario principal.
7. Sélectionnez le symbole de clip dans la bibliothèque et choisissez Liaison dans le menu contextuel Bibliothèque.

   La boîte de dialogue Propriétés de liaison apparaît.
8. Choisissez l'option Exporter pour ActionScript , puis Exporter dans la première image.
9. Saisissez **dynamic_id**dans la zone de texte de l'identifiant, puis cliquez sur OK.
10. Sélectionnez la première image du scénario principal et ajoutez le code suivant dans la fenêtre de script du panneau Actions :

    ```
    /* Joint un nouveau clip et le déplace aux coordonnées x et y de 50 */
    this.attachMovie("dynamic_id", "newClip_mc", 99, {name_str:"Erick",
      _x:50, _y:50});
    ```
11. Testez le document Flash (Contrôle > Tester animation).

    Le nom que vous avez spécifié dans l'appel `attachMovie()` apparaît dans le champ de texte du nouveau clip.

Pour un exemple de fichier source, gallery_tween.fla, qui décrit l'utilisation du code ActionScript pour contrôler les clips dynamiquement au cours du chargement de fichiers images dans un fichier SWF et qui inclut la possibilité de rendre chaque clip déplaçable, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Galleries afin d'accéder à l'exemple.

Pour un exemple de fichier source, animation.fla, qui crée et supprime de nombreux clips à l'exécution, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Animation afin d'accéder à l'exemple.

Gestion des profondeurs de clip

Chaque clip possède son propre ordre *z* qui détermine la façon dont les objets se chevauchent à l'intérieur du fichier SWF parent ou du clip. Chaque clip est associé à une valeur de profondeur qui détermine s'il sera rendu devant ou derrière les autres clips dans le même scénario de clip. Lorsque vous créez un clip lors de l'exécution à l'aide de `attachMovie (méthode MovieClip.attachMovie)`, `duplicateMovieClip (méthode MovieClip.duplicateMovie)` ou `createEmptyMovieClip (méthode MovieClip.createEmptyMovieClip)`, vous pouvez toujours spécifier une profondeur pour le nouveau clip en tant que paramètre de méthode. Par exemple, le code suivant associe un nouveau clip au scénario d'un clip nommé `conteneur_mc` avec une valeur de profondeur de10.

```
container_mc.attachMovie("symbolID", "clip1_mc", 10);
```

Cet exemple crée un nouveau clip avec une profondeur de 10 dans l'espace d'ordre *z* de `container_mc`.

Le code suivant associe deux nouveaux clips à `container_mc`. Le premier clip, nommé `clip1_mc`, sera rendu derrière `clip2_mc`, car une valeur de profondeur inférieure lui a été attribuée.

```
container_mc.attachMovie("symbolID", "clip1_mc", 10);
container_mc.attachMovie("symbolID", "clip2_mc", 15);
```

Les valeurs de profondeur pour les clips peuvent aller de -16384 à 1048575. Si vous créez ou associez un nouveau clip à une profondeur déjà associée à un autre clip, le nouveau clip ou clip associé remplace le contenu existant. Pour éviter ce problème, utilisez la méthode `MovieClip.getNextHighestDepth()` ; toutefois, n'employez pas cette méthode avec des composants qui utilisent un système différent de gestion de profondeur. Utilisez plutôt la classe DepthManager avec des occurrences de composant. Pour plus d'informations, consultez le *Guide de référence du langage ActionScript 2.0*.

La classe MovieClip offre plusieurs méthodes de gestion de la profondeur des clips. Pour plus d'informations, consultez les sections `getNextHighestDepth (méthode MovieClip.getNextHighestDepth)`, `getInstanceAtDepth (méthode MovieClip.getInstanceAtDepth)`, `getDepth (méthode MovieClip.getDepth)` et `swapDepths (méthode MovieClip.swapDepths)` dans le *Guide de référence du langage ActionScript 2.0*.

Pour plus d'informations sur les profondeurs de clip, consultez les rubriques suivantes :

- « Définition de la profondeur maximale disponible suivante », à la page 351
- « Définition de l'occurrence à une profondeur spécifique », à la page 351
- « Définition de la profondeur d'une occurrence », à la page 352
- « Permutation des profondeurs de clips », à la page 352

Définition de la profondeur maximale disponible suivante

Pour déterminer quelle est la profondeur maximale disponible suivante dans un clip, utilisez `MovieClip.getNextHighestDepth()`. La valeur entière renvoyée par cette méthode indique la prochaine profondeur disponible qui sera rendue devant tous les autres objets dans le clip.

Le code suivant associe un nouveau clip, avec une valeur de profondeur de 10, sur le scénario racine nommé `file_mc`. Il détermine ensuite la prochaine profondeur maximale disponible dans le même clip, puis crée un nouveau clip, `edit_mc`, à cette profondeur.

```
this.attachMovie("menuClip","file_mc", 10, {_x:0, _y:0});
trace(file_mc.getDepth()); // 10
var nextDepth:Number = this.getNextHighestDepth();
this.attachMovie("menuClip", "edit_mc", nextDepth, {_x:200, _y:0});
trace(edit_mc.getDepth()); // 11
```

Dans ce cas, la variable nommée `nextDepth` contient la valeur 11 puisqu'il s'agit de la prochaine profondeur maximale disponible pour le clip `edit_mc`.

N'utilisez pas `MovieClip.getNextHighestDepth()` avec des composants ; préférez-lui le gestionnaire de profondeur. Pour plus d'informations, consultez la « Classe DepthManager » dans la *Référence du langage des composants ActionScript 2.0*. Pour plus d'informations sur `MovieClip.getNextHighestDepth()`, consultez la section `getNextHighestDepth (méthode MovieClip.getNextHighestDepth)`..

Pour obtenir la profondeur maximale actuellement occupée, soustrayez 1 à la valeur renvoyée par `getNextHighestDepth()`, comme illustré dans la section suivante.

Définition de l'occurrence à une profondeur spécifique

Pour définir l'occurrence à une profondeur spécifique, utilisez `MovieClip.getInstanceAtDepth()`. Cette méthode renvoie une référence à l'occurrence de MovieClip se trouvant à la profondeur indiquée.

Le code suivant combine `getNextHighestDepth()` et `getInstanceAtDepth()` pour déterminer le clip se trouvant à la plus grande profondeur (actuellement) occupée du scénario racine.

```
var highestOccupiedDepth:Number = this.getNextHighestDepth() - 1;
var instanceAtHighestDepth:MovieClip =
  this.getInstanceAtDepth(highestOccupiedDepth);
```

Pour plus d'informations, consultez `getInstanceAtDepth (méthode MovieClip.getInstanceAtDepth)` dans le *Guide de référence du langage ActionScript 2.0*.

Définition de la profondeur d'une occurrence

Pour déterminer la profondeur d'une occurrence de clip, utilisez `MovieClip.getDepth()`.

Le code suivant itère sur tous les clips du scénario principal d'un fichier SWF et affiche le nom d'occurrence et la valeur de profondeur de chaque clip dans le panneau de sortie :

```
for (var item:String in _root) {
  var obj:Object = _root[item];
  if (obj instanceof MovieClip) {
    var objDepth:Number = obj.getDepth();
    trace(obj._name + ":" + objDepth)
  }
}
```

Pour plus d'informations, consultez `getDepth (méthode MovieClip.getDepth)` dans le *Guide de référence du langage ActionScript 2.0*.

Permutation des profondeurs de clips

Pour permuter les profondeurs de deux clips sur un même scénario, utilisez `MovieClip.swapDepths()`. Les exemples suivants montrent comment deux occurrences de clips peuvent permuter leurs profondeurs en cours d'exécution.

Pour permuter des profondeurs de clips :

1. Créez un document Flash appelé **swap.fla**.
2. Dessinez un cercle bleu sur la scène.
3. Sélectionnez le cercle bleu, puis Modifier > Convertir en symbole.
4. Sélectionnez l'option clip et cliquez sur « OK ».
5. Sélectionnez l'occurrence sur la scène, puis tapez **first_mc** dans la zone de texte Nom de l'occurrence dans l'inspecteur des propriétés.
6. Dessinez un cercle rouge sur la scène, puis choisissez Modifier > Convertir en symbole.
7. Sélectionnez l'option clip et cliquez sur OK.
8. Sélectionnez l'occurrence sur la scène, puis tapez **second_mc** dans la zone de texte Nom de l'occurrence dans l'inspecteur des propriétés.
9. Faites glisser les deux occurrences de manière à ce qu'elles chevauchent légèrement la scène.

10. Sélectionnez la première image du scénario, puis ajoutez le code suivant au panneau Actions :

```
first_mc.onRelease = function() {
     this.swapDepths(second_mc);
  };
  second_mc.onRelease = function() {
    this.swapDepths(first_mc);
  };
```

11. Sélectionnez Contrôle > Tester l'animation pour tester le document.

    Lorsque vous sélectionnez les occurrences sur la scène, elles permutent leurs profondeurs Vous verrez que les deux occurrences changent la position des clips l'un au-dessus de l'autre.

Pour plus d'informations, consultez `swapDepths (méthode MovieClip.swapDepths)` dans le *Guide de référence du langage ActionScript 2.0*.

Mise en cache et parcours de clips à l'aide d'ActionScript

Que vous créiez une application ou des animations scriptées complexes, vous devez considérer la performance et l'optimisation, à mesure que la taille de vos conceptions Flash augmente. Lorsque votre contenu demeure statique (par exemple un clip rectangle), Flash ne l'optimise pas. Toutefois, lorsque vous modifiez la position du clip rectangle, Flash redessine la totalité du rectangle dans Flash Player 7 et les versions antérieures.

Dans Flash 8, vous pouvez mettre certains clips et boutons en cache pour améliorer les performances de votre fichier SWF. Le clip ou bouton est une *surface*, c'est-à-dire la version bitmap des données vectorielles de l'occurrence, données non destinées à être énormément modifiées tout au long de la vie de votre fichier SWF. Ainsi, les occurrences pour lesquelles la mise en cache est activée ne sont pas continuellement redessinées pendant la lecture du fichier SWF, et le rendu de ce dernier est rapide.

REMARQUE	Vous pouvez mettre à jour les données vectorielles au moment de la recréation de la surface. Ainsi, les données vectorielles mises en cache dans la surface ne doivent pas nécessairement être les mêmes pour l'ensemble du fichier SWF.

Vous pouvez utiliser ActionScript pour activer la mise en cache, le parcours et le contrôle des arrière-plans. Vous pouvez activer la mise en cache pour une occurrence de clip dans l'inspecteur des propriétés. Pour mettre en cache des clips ou des boutons sans l'aide d'ActionScript, vous pouvez en revanche sélectionner l'option d'utilisation de la mise en cache bitmap en cours d'exécution dans l'inspecteur des propriétés.

Le tableau suivant contient une brève description des nouvelles propriétés d'occurrences de clips :

Propriété	Description
cacheAsBitmap	L'occurrence de clip met en cache une représentation bitmap d'elle-même. Flash crée un objet de surface pour l'occurrence, correspondant à une image bitmap mise en cache et non à des données vectorielles. Si vous modifiez les limites du clip, la surface est recréée et non redimensionnée. Pour plus d'informations et un exemple, consultez la section « Mise en cache d'un clip », à la page 357.
opaqueBackground	L'occurrence de clip est opaque et vous pouvez définir la couleur d'arrière-plan. Si vous définissez cette propriété sur une valeur numérique, la surface de l'occurrence de clip est opaque (non transparente). Une image bitmap opaque n'a pas de canal alpha (transparence) et son rendu est plus rapide. Pour plus d'informations et un exemple, consultez la section « Réglage de l'arrière-plan d'un clip », à la page 360.
scrollRect	Cette propriété permet de parcourir rapidement le contenu du clip et d'ouvrir une fenêtre capable d'afficher davantage de contenu. Le contenu du clip est recadré et l'occurrence défile avec une largeur, une longueur et des décalages de défilement spécifiés. L'utilisateur peut ainsi parcourir rapidement le contenu du clip et ouvrir une fenêtre plus grande que la scène. Les champs de texte et le contenu complexe affichés dans l'occurrence peuvent défiler plus rapidement car Flash ne régénère pas l'ensemble des données vectorielle du clip. Pour plus d'informations et un exemple, consultez la section `scrollRect (propriété MovieClip.scrollRect)`.

Ces trois propriétés sont indépendantes les unes des autres, toutefois les propriétés `opaqueBackground` et `scrollRect` fonctionnent mieux quand un objet est caché en tant que bitmap. Vous ne constaterez les bienfaits en performance pour les propriétés `opaqueBackground` et `scrollRect` que lorsque vous réglerez `cacheAsBitmap` sur `true`.

Pour créer une surface qui puisse défiler, vous devez définir les propriétés `cacheAsBitmap` et `scrollRect` pour l'occurrence de clip. Les surfaces peuvent s'imbriquer dans d'autres surfaces. La surface copiera l'image bitmap sur sa surface parent.

Pour des informations sur le masquage du canal alpha, qui exige que vous définissiez la propriété du `cacheAsBitmap` sur `true`, consultez « Masquage du canal alpha », à la page 362.

REMARQUE

Vous ne pouvez pas appliquer la mise en cache directement aux champs de texte. Pour profiter de cette fonctionnalité, vous devez placer le texte dans un clip. Pour un exemple de fichier, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/CacheBitmap afin d'accéder à l'exemple.

Pour des exemples sur l'application de mises en cache bitmap dans une occurrence et dans du texte de défilement, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Les exemples suivants sont disponibles :

- cacheBitmap.fla ; Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/CacheBitmap.
- aliasing.fla ; Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Advanced Anti-Aliasing.

Quand activer la mise en cache

L'activation de la mise en cache pour un clip crée une surface dont les avantages sont multiples, par exemple pour accélérer le rendu des animations vectorielles complexes. Lorsque vous souhaitez activer la mise en cache, plusieurs scénarios sont disponibles. Il pourrait sembler avantageux de toujours activer la mise en cache pour améliorer les performances de votre fichier SWF. Cependant, dans certains cas, cette opération ne les améliore pas, voire même les diminue. Cette section présente des scénarios où la mise en cache s'avère bénéfique et d'autres où il est préférable d'utiliser des clips ordinaires.

Les performances générales des données mises en cache dépendront de la complexité des données vectorielles de vos occurrences, de la quantité de modifications et de la définition, ou non, de la propriété `opaqueBackground`. Si vous modifiez de petites zones, la différence entre l'utilisation d'une surface et celle de données vectorielles sera négligeable. Vous pouvez dans ce cas tester les deux scénarios avant de déployer votre application.

Pour des informations sur le masquage du canal alpha, qui exige que vous définissiez la propriété du `cacheAsBitmap` sur `true`, consultez « Masquage du canal alpha », à la page 362.

Quand utiliser le cache bitmap

Ce qui suit est une série de scénarios dans lesquels vous pouvez voir les bénéfices significatifs qui résultent de la mise en cache bitmap.

Image complexe d'arrière plan Une application qui contient une image de fond complexe de données vectorielles (peut-être une image à laquelle vous avez appliqué la commande de traçage de bitmap ou une création artistique créée dans Adobe Illustrator). Vous pouvez animer les caractères sur l'arrière-plan, ce qui ralentit l'animation parce que l'arrière-plan a besoin de continuellement régénérer les données vectorielles. Pour améliorer les performances, vous pouvez sélectionner le contenu, le stocker dans un clip et définir la propriété `opaqueBackground` sur `true`. L'arrière-plan est rendu en tant que bitmap et peut être redessiné rapidement pour que l'animation se joue beaucoup plus vite.

Champ de texte de défilement Application qui affiche une grande quantité de texte dans une zone de texte de défilement. Vous pouvez placer la zone de texte dans un clip que vous définissez comme défilant à l'aide de bornes de défilement (propriété `scrollRect`). Ceci permet un déroulement de pixels rapide pour l'occurrence donnée. Quand un utilisateur déroule l'occurrence de clip, Flash fait défiler les pixels déroulés vers le haut et génère la zone nouvellement exposée au lieu de régénérer toute la zone de texte.

Système de fenêtres Application comportant un système complexe de chevauchement de fenêtres. Chaque fenêtre peut être ouverte ou fermée (par exemple, les fenêtres de navigateur Web). Si vous marquez chaque fenêtre en tant que surface (définissez la propriété `cacheAsBitmap` sur `true`), chaque fenêtre sera isolée et cachée. Les utilisateurs peuvent faire glisser les fenêtres de manière à ce qu'elles se chevauchent. Chaque fenêtre n'a pas besoin de regénérer le contenu vectoriel.

Tous ces scénarios améliorent la réactivité et l'interactivité de l'application en optimisant les graphiques vectoriels.

Pour des exemples sur l'application de mises en cache bitmap dans une occurrence et dans du texte de défilement, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Les exemples suivants sont disponibles :

- cacheBitmap.fla ; Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/CacheBitmap.
- aliasing.fla ; Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Advanced Anti-Aliasing.

Quand éviter d'utiliser la mise en cache bitmap

Un mauvais usage de cette fonctionnalité peut avoir un effet négatif sur votre fichier SWF. Lorsque vous créez un fichier FLA qui utilise des surfaces, gardez à l'esprit les points suivants :

- N'abusez pas des surfaces (clips avec mise en cache activée). Chaque surface utilise plus de mémoire qu'un clip courant, ce qui signifie que vous ne devez activer les surfaces que quand vous devez améliorer les performances de rendu.

  Un bitmap caché utilise beaucoup plus de mémoire qu'une occurrence de clip courant Par exemple, si le clip sur la scène a une taille de 250 pixels sur 250 pixels, il peut occuper 250 KO s'il est caché, contre 1 KO s'il est courant (non caché).
- Évitez de zoomer dans les surfaces cachées. Si vous abusez de la mise en cache bitmap, une grande quantité de mémoire sera occupée (voir la puce précédente), surtout si vous zoomez sur le contenu.
- Utilisez des surfaces pour les occurrences de clip qui sont grandement statiques (non-animées). Vous pouvez faire glisser ou déplacer l'occurrence, mais son contenu ne doit pas être animé ni subir de nombreuses modifications. Par exemple, si vous faites pivoter ou si vous transformez une occurrence, celle-ci change entre la surface et les données vectorielles, ce qui rend le traitement difficile et endommage votre fichier SWF.
- Si vous m^lez des surfaces à des données vectorielles, cela augmente la quantité de traitement que Flash Player (et parfois l'ordinateur) doit accomplir. Rassemblez les surfaces autant que possible ; par exemple, lorsque vous créez des applications de fenêtrage.

Mise en cache d'un clip

Pour mettre en cache une occurrence de clip, il faut définir la propriété `cacheAsBitmap` sur `true`. Après avoir défini la propriété `cacheAsBitmap` sur `true`, vous remarquerez que l'occurrence de clip accroche les pixels sur toutes les coordonnées. Lorsque vous testez le fichier SWF, vous devriez remarquer que le rendu des animations vectorielles complexes est bien plus rapide.

Une surface (bitmap caché) n'est pas créée même quand `cacheAsBitmap` est défini sur `true`, si l'un ou l'autre des événements suivants survient :

- Le bitmap fait plus de 2 880 pixels en hauteur ou en largeur.
- Le bitmap ne peut être alloué en mémoire (erreur de manque de mémoire).

Mise en cache d'un clip

1. Créez un nouveau document Flash, puis nommez le fichier **cachebitmap.fla**.
2. Tapez **24** dans la case de texte fps dans l'inspecteur des propriétés (Fenêtre > Propriétés > Propriétés).
3. Créez ou importez un graphique vectoriel complexe dans le fichier FLA.

   Pour un exemple de source de graphique vectoriel complexe, CacheBitmap, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/CacheBitmap afin d'accéder à l'exemple.
4. Sélectionnez le graphique vectoriel, puis Modifier > Concertir en symbole.
5. Tapez **star** dans la case de texte Nom, puis cliquez sur Avancé (si la boîte de dialogue n'est pas déjà développée).
6. Sélectionnez Exporter pour ActionScript (ce qui sélectionne également Exporter dans la première image).
7. Tapez **star_id** dans la case de texte de l'identifiant.
8. Cliquez sur « OK » pour créer le symbole du clip, avec la liaison identifiante de Star.
9. Sélectionnez la première image du scénario, puis ajoutez le code ActionScript suivant au panneau Actions :

```
import mx.transitions.Tween;

var star_array:Array = new Array();
for (var i:Number = 0; i < 20; i++) {
  makeStar();
}
function makeStar():Void {
  var depth:Number = this.getNextHighestDepth();
  var star_mc:MovieClip = this.attachMovie("star_id", "star" + depth,
  depth);
  star_mc.onEnterFrame = function() {
    star_mc._rotation += 5;
  }
  star_mc._y = Math.round(Math.random() * Stage.height - star_mc._height
  / 2);
  var star_tween:Tween = new Tween(star_mc, "_x", null, 0, Stage.width,
  (Math.random() * 5) + 5, true);
  star_tween.onMotionFinished = function():Void  {
    star_tween.yoyo();
  };
  star_array.push(star_mc);
}
```

```
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function():Void {
  var star_mc:MovieClip;
  for (var i:Number = 0; i < star_array.length; i++) {
    star_mc = star_array[i];
    star_mc.cacheAsBitmap = !star_mc.cacheAsBitmap;
  }
}
Mouse.addListener(mouseListener);
```

10. Sélectionnez Contrôle > Tester l'animation pour tester le document.
11. Cliquez n'importe où sur la scène pour activer la mise en cache bitmap.

    Vous remarquerez que l'animation apparaît puis s'anime à une image par seconde et passe à une animation fluide où les occurrences s'animent de long en large à travers la scène. Quand vous cliquez sur la scène, cela fait basculer les réglages `cacheAsBitmap` entre `true` et `false`.

Si vous activez et désactivez la mémoire cache comme illustré dans l'exemple précédent, cela libère les données mises en cache. Vous pouvez également appliquer ce code pour une occurrence de bouton. Consultez `cacheAsBitmap(propriété Button.cacheAsBitmap)` dans le *Guide de référence du langage ActionScript 2.0*.

Pour plus d'informations sur les clips défilants, consultez `scrollRect(propriété MovieClip.scrollRect)` dans le *Guide de référence du langage ActionScript 2.0.* Pour des informations sur le masquage du canal alpha, qui demande que vous définissiez la propriété `cacheAsBitmap` sur `true`, consultez « Masquage du canal alpha », à la page 362.

Pour des exemples sur l'application de mises en cache bitmap dans une occurrence et dans du texte de défilement, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Les exemples suivants sont disponibles :

- cacheBitmap.fla ; Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/CacheBitmap.
- aliasing.fla ; Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Advanced Anti-Aliasing.

Réglage de l'arrière-plan d'un clip

Vous pouvez définir un arrière-plan opaque pour un clip. Par exemple, quand vous avez un arrière-plan qui contient un art vectoriel complexe, vous pouvez définir la propriété `opaqueBackground` sur une couleur donnée (en général la même couleur que la scène). L'arrière-plan est alors considéré comme un bitmap, ce qui permet d'optimiser les performances.

Quand vous définissez `cacheAsBitmap` sur `true` et la propriété `opaqueBackground` sur une couleur donnée, la propriété d'arrière-plan opaque permet que le bitmap interne soit opaque et rendu plus rapidement. Si vous ne définissez pas `cacheAsBitmap`sur true, la propriété `opaqueBackground`ajoute une forme carrée vectorielle opaque à l'arrière-plan de l'occurrence de clip. Cela ne crée par un bitmap automatiquement.

L'exemple suivant décrit comment définir l'arrière-plan d'un clip pour optimiser les performances.

Pour définir l'arrière-plan d'un clip :

1. Créez un document Flash et enregistrez-le sous le nom **background.fla**.
2. Dessinez un cercle bleu sur la scène.
3. Sélectionnez le cercle bleu, puis Modifier > Convertir en symbole.
4. Sélectionnez l'option clip et cliquez sur OK.
5. Sélectionnez l'occurrence sur la scène, puis tapez **my_mc** dans la case de texte Nom de l'occurence dans l'inspecteur des propriétés.
6. Sélectionnez Image 1 du scénario, puis ajoutez le code suivant au panneau Actions :

```
/* Quand vous définissez cacheAsBitmap, le bitmap interne est opaque et
  son rendu est plus rapide. */
my_mc.cacheAsBitmap = true;
my_mc.opaqueBackground = 0xFF0000;
```

7. Sélectionnez Contrôle > Tester l'animation pour tester le document.

   Le clip apparaît sur la scène avec la couleur de l'arrière-plan spécifiée.

Pour plus d'informations sur cette propriété, consultez `opaqueBackground(propriété MovieClip.opaqueBackground)` dans le *Guide de référence du langage ActionScript 2.0* .

Utilisation des clips en tant que masques

Vous pouvez utiliser un clip comme masque pour créer un trou qui laisse apparaître le contenu d'un autre clip. Le clip masque joue toutes les images dans son scénario comme dans un clip courant. Vous pouvez rendre le clip déplaçable, l'animer le long d'un guide de mouvement, utiliser des formes distinctes dans un même masque, ou redimensionner un masque de façon dynamique. Vous pouvez également utiliser ActionScript pour activer ou désactiver un masque.

Il est impossible d'utiliser un masque pour en masquer un autre. Il est impossible de définir la propriété `_alpha` d'un clip utilisé comme masque. Seuls les remplissages sont utilisés dans un clip employé comme masque ; les traits sont ignorés.

Pour créer un masque :

1. Créez un carré sur la scène à l'aide de l'outil Rectangle.
2. Sélectionnez le carré et appuyez sur F8 pour le convertir en clip.

   Cette occurrence est votre masque.
3. Dans l'inspecteur des propriétés, tapez **mask_mc** dans la case de texte Nom de l'occurrence.

   Le clip masqué se joue sous toutes les zones opaques (non transparentes) du clip agissant comme le masque.
4. Sélectionnez l'image 1 dans le scénario.
5. Ouvrez le panneau Actions (Fenêtre > Actions) si ce n'est pas déjà fait.
6. Dans le panneau Actions, entrez le code suivant :

```
System.security.allowDomain("http://www.helpexamples.com");

this.createEmptyMovieClip("img_mc", 10);
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip):Void {
  target_mc.setMask(mask_mc);
}
var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.addListener(mclListener);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);
```

7. Sélectionnez Contrôle > Tester l'animation pour tester le document.

   Une image JPEG externe se charge dans le fichier SWF en cours d'exécution et sera masquée par la forme précédemment dessinée sur la scène.

Pour plus d'informations, consultez setMask (`méthode MovieClip.setMask`) dans le *Guide de référence du langage ActionScript 2.0*.

Polices de périphérique des masques

Vous pouvez utiliser un clip pour masquer le texte défini dans une police de périphérique. Pour qu'un masque clip fonctionne correctement sur une police de périphérique, l'utilisateur doit posséder Flash Player 6 (6.0.40.0) ou une version ultérieure.

Lorsque vous utilisez un clip pour masquer le texte défini dans une police de périphérique, le cadre de délimitation rectangulaire du masque est utilisé comme forme de masque. Ainsi, si vous créez un masque clip non rectangulaire pour un texte de police de périphérique dans l'environnement autorisant Flash, le masque qui apparaît dans le fichier SWF est la forme de la boîte de délimitation rectangulaire du masque, et non celle du masque en lui-même.

Vous pouvez uniquement masquer des polices de périphérique en utilisant un clip comme masque. Vous ne pouvez pas masquer des polices de périphérique en utilisant un calque de masque sur la scène.

Masquage du canal alpha

Le masquage du canal alpha est pris en charge si le masque et les clips masques utilisent la mise en cache bitmap. Cette prise en charge permet également d'utiliser un filtre sur le masque indépendamment du filtre qui est appliqué sur le clip masque lui-même.

Pour obtenir un exemple de masquage alpha, téléchargez le fichier exemple de masquage alpha depuis le site www.adobe.com/go/learn_fl_samples_fr.

Dans ce fichier exemple, le masque est oval (`oval_mask`) qui possède un alpha de 50% et un filtre de flou lui est appliqué. Le clip masque (`flower_maskee`) possède un alpha de 100% et aucun filtre ne lui est appliqué. Les deux clips possèdent une mise en cache bitmap en cours appliquée dans l'inspecteur des propriétés.

Dans le panneau Actions, le code suivant est placé sur l'image 1 du scénario :

```
flower_maskee.setMask(oval_mask);
```

Quand vous testez le document (Contrôle > Tester l'animation), le clip masque est mêlé d'alpha en raison de l'emploi du masque.

REMARQUE

Les couches de masques ne prennent pas en charge le masquage du canal alpha. Vous devez utiliser le code ActionScript pour applicquer un masque et utiliser la mise en cache bitmap pour le temps d'exécution.

Gestion des événements clip

Les clips peuvent répondre à des événements utilisateur, tels que des clics de souris ou des pressions sur des touches, ainsi qu'à des événements de niveau système, tel que le chargement initial d'un clip sur la scène. ActionScript fournit deux façons de gérer les événements de clip : à travers les méthodes de gestion d'événements et les gestionnaires d'événements `onClipEvent()` et `on()`. Pour plus d'informations sur la gestion d'événements clip, consultez le Chapitre 9, « Gestion d'événements ».

Affectation d'une classe à un symbole de clip

En utilisant ActionScript 2.0, vous pouvez créer une classe qui élargit le comportement de la classe de clip incorporée, puis utiliser la boîte de dialogue des propriétés de liaison pour affecter cette classe à un symbole de bibliothèque d'un clip. Lorsque vous créez une occurrence du clip auquel la classe est affectée, elle assume les propriétés et comportements définis par la classe qui lui est affectée. (Pour plus d'informations sur ActionScript 2.0, consultez « Exemple : Ecriture de classes personnalisées », à la page 238).

Dans une sous-classe de la classe MovieClip, vous pouvez fournir les définitions des méthodes et gestionnaires d'événement intégrés MovieClip, tels que `onEnterFrame` et `onRelease`. Dans la procédure suivante, vous créez une classe intitulée MoveRight qui étend la classe MovieClip et définit un gestionnaire `onPress` qui déplace le clip de 20 pixels vers la droite chaque fois que l'utilisateur clique sur le clip. Dans la deuxième procédure, vous créerez un symbole de clip dans un nouveau document Flash (FLA) et affecterez la classe MoveRight à ce symbole.

Pour créer une sous-classe de clip :

1. Créez un répertoire et enregistrez-le sous le nom **BallTest**.
2. Choisissez Fichier > Nouveau, puis sélectionnez un fichier ActionScript dans la liste des types de document pour créer un nouveau fichier ActionScript.
3. Entrez le code suivant dans votre fichier de script :

```
// Classe MoveRight -- déplace le clip vers la droite suite à un clic
class MoveRight extends MovieClip {
  public function onPress() {
    this._x += 20;
  }
}
```

4. Enregistrez le document en tant que MoveRight.as dans le répertoire BallTest.

Pour affecter la classe à un symbole de clip :

1. Dans Flash, choisissez Fichier > Nouveau, sélectionnez Document Flash dans la liste des types de fichiers et cliquez sur OK.
2. A l'aide de l'outil Ovale, dessinez un cercle sur la scène.
3. Sélectionnez le cercle puis choisissez Modification > Convertir en symbole.
4. Dans la boîte de dialogue de conversion en symbole, sélectionnez Clip comme le comportement du symbole et saisissez **ball_mc** dans la case de texte Nom.
5. Sélectionnez Avancé pour afficher les options de Liaison, si elles ne sont pas visibles.
6. Sélectionnez l'option d'exportation pour ActionScript et tapez **mask_mc** dans la case de texte de la classe. Cliquez sur OK.
7. Enregistrez le fichier sous ball.fla dans le répertoire BallTest (le répertoire contenant le fichier MoveRight.as).
8. Testez le document Flash (Contrôle > Tester l'animation).

   Chaque fois que vous cliquez sur le clip ball, il se déplace de 20 pixels sur la droite.

Si vous créez des propriétés de composante pour une classe et si vous souhaitez qu'un clip hérite de ces propriétés de composante, il faut passer à l'étape suivante : Après avoir sélectionné le symbole de clip dans le panneau Bibliothèque, sélectionnez Définition de composante depuis le menu contextuel de la Bibliothèque et entrez le nouveau nom de classe dans la boîte Classe.

Initialisation des propriétés de classe

Dans l'exemple présenté lors de la seconde procédure sous « Affectation d'une classe à un symbole de clip », vous avez ajouté l'occurrence du symbole de la Balle sur la scène pendant la programmation. Comme présenté dans « Ajout de paramètres aux clips créés dynamiquement », à la page 348, vous pouvez affecter des paramètres à des clips que vous créez pour le temps d'exécution en utilisant le paramètre *`initObject`* de `attachMovie()` et `duplicateMovie()`. Vous pouvez utiliser cette fonctionnalité pour initialiser des propriétés de la classe que vous affectez à un clip.

Par exemple, la classe suivante appelée MoveRightDistance est une variante de la classe MoveRight (consultez « Affectation d'une classe à un symbole de clip », à la page 363). La différence est une nouvelle propriété nommée `distance`, dont la valeur détermine le nombre de pixels dont un clip se déplace chaque fois que vous cliquez dessus.

Pour transmettre des arguments à une classe personnalisée :

1. Créez un nouveau document ActionScript, puis enregistrez-le sous le nom de **MoveRightDistance.as**.
2. Tapez le code ActionScript suivant dans la fenêtre Script :

```
// classe MoveRightDistance -- déplace le clip vers la droite dans
// chaque image.
class MoveRightDistance extends MovieClip {
  // La propriété Distance détermine de combien de
  // pixels déplacer le clip à chaque clic de la souris.
  var distance:Number;
  function onPress() {
    this._x += this.distance;
  }
}
```

3. Enregistrez votre progression.
4. Créez un document Flash, puis enregistrez-le sous le nom de **MoveRightDistance.fla** dans le même répertoire que le fichier de classe.
5. Créez un symbole de clip qui contient une forme vectorielle, telle qu'une forme ovale, puis supprimez tout le contenu de la scène.

   Il vous suffit d'un symbole de clip dans la bibliothèque pour cet exemple.
6. Dans le panneau Bibliothèque, cliquez-droit (Windows) ou tapez Contrôle-clic (Macintosh) sur le symbole et choisissez Liaison dans le menu contextuel.
7. Affectez l'identifiant de liaison **Ball** au symbole.
8. Tapez **MoveRightDistance** dans la case de texte de la classe AS 2.0.
9. Ajoutez le code suivant à l'image 1 du scénario :

```
this.attachMovie("Ball", "ball50_mc", 10, {distance:50});
this.attachMovie("Ball", "ball125_mc", 20, {distance:125});
```

   Ce code crée deux occurrences du symbole sur le scénario racine du fichier SWF. La première occurrence, nommée `ball50_mc`, se déplace de 50 pixels à chaque fois qu'on clique dessus ; la seconde, nommée `ball125_mc`, se déplace de 125 pixels à chaque fois qu'on clique dessus.
10. Choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

CHAPITRE 11

Utilisation du texte et des chaînes

11

Un grand nombre des applications, présentations et graphiques que vous créez à l'aide de Flash contiennent du texte. Vous pouvez utiliser toutes sortes de textes. Vous pourriez utiliser un texte statique dans vos présentations et un texte dynamique pour les passages plus longs. Vous pouvez également utiliser la saisie de texte pour capturer la saisie de l'utilisateur, et du texte dans une image pour votre motif d'arrière-plan. Il est possible de créer des champs de texte à l'aide de l'outil de programmation Flash ou d'utiliser ActionScript.

Pour afficher un texte, une solution consiste à utiliser un code pour manipuler l'aspect des chaînes avant qu'elles ne soient chargées et affichées sur la scène au moment de l'exécution. Il existe plusieurs moyens de manipuler les chaînes dans une application : on peut par exemple les envoyer vers un serveur et récupérer une réponse, analyser les chaînes dans un tableau ou valider les chaînes que l'utilisateur saisit dans un champ de texte.

Ce chapitre décrit plusieurs manières d'utiliser le texte et les chaînes dans vos applications en mettant l'accent sur l'utilisation du code pour manipuler le texte.

La liste suivante décrit la terminologie utilisée dans ce chapitre.

Alias Le texte aliasé n'utilise pas de variations de couleur pour que ses bordures découpées semblent plus lisses, contrairement au texte anti-aliasé (voir la définition suivante).

Anti-alias Vous utiliserez l'anti-alias pour lisser le texte afin que les bords des caractères apparaissant à l'écran semblent moins accentués. L'option Anti-alias de Flash rend le texte plus lisible en alignant le contour du texte le long des limites de pixels et est particulièrement efficace pour rendre nettement les petites polices.

Caractères Les caractères sont des lettres, des chiffres et des signes de ponctuation que vous combinez pour former des chaînes.

Polices de périphérique Les polices de périphérique sont des polices spéciales de Flash qui ne sont pas incorporées à un fichier SWF. Flash Player utilise en fait la police présente sur l'ordinateur local qui ressemble le plus à la police de périphérique. Dans la mesure où le contour des polices n'est pas incorporé, la taille d'un fichier SWF est inférieure que si le contour des polices est incorporé. Toutefois, dans la mesure où les polices de périphérique ne sont pas incorporées, le texte que vous créez avec ces polices semble différent de ce que vous pouvez attendre sur les systèmes informatiques ne disposant pas d'une police installée correspondant à la police de périphérique. Flash intègre trois polices de périphérique : _sans (analogue à Helvetica et Arial), _serif (analogue à Times Roman) et _typewriter (analogue à Courier).

Polices Jeux de caractères présentant un aspect, un style et une taille analogues.

Chaîne Séquence de caractères.

Texte Série d'une ou plusieurs chaînes pouvant être affichées dans un champ de texte ou au sein d'un composant d'interface utilisateur.

Champs de texte Elément visuel de la scène qui vous permet d'afficher du texte pour un utilisateur. Tout comme le contrôle des formes d'un champ de saisie de texte ou d'une zone de texte en HTML, Flash vous permet de définir les champs de texte comme modifiables (lecture seule), d'autoriser le formatage HTML, d'activer la prise en charge multiligne, de masquer le mot de passe ou d'appliquer une feuille de style CSS à votre texte au format HTML.

Formatage de texte Vous pouvez appliquer le formatage à un champ de texte ou à certains caractères d'un champ de texte. Voici certains exemples d'options de formatage de texte pouvant être appliquées à du texte : alignement, retrait, gras, couleur, taille de la police, largeur des marges, italiques et espacement des lettres.

Pour plus d'informations sur le texte, consultez les rubriques suivantes :

A propos des champs de texte

Un champ de texte dynamique ou de saisie est un objet TextField (occurrence de la classe TextField). Lorsque vous créez un champ de texte dans l'environnement de programmation, vous pouvez lui affecter un nom d'occurrence dans l'inspecteur des propriétés. Utilisez ce nom d'occurrence dans les instructions ActionScript pour définir, modifier et formater le champ de texte et son contenu à l'aide des classes TextField et TextFormat.

Vous pouvez utiliser l'interface utilisateur pour créer différents champs de texte, ou utiliser ActionScript pour créer des champs de texte. Vous pouvez créer les types de champs de texte suivants dans Flash :

Texte statique Utilisez le texte statique pour afficher les caractères n'ayant pas à être modifiés, afficher de petites quantités de texte ou des polices spéciales non disponibles sur la plupart des ordinateurs. Vous pouvez également afficher les polices rares en incorporant les caractères destinés à des champs de texte dynamiques.

Texte dynamique Utilisez les champs de texte dynamique pour afficher des caractères mis à jour ou modifiés au moment de l'exécution. Vous pouvez également charger du texte dans des champs de texte dynamiques.

Texte de saisie Les champs de texte de saisie permettent de récupérer les entrées des utilisateurs. Ces derniers peuvent saisir du texte dans ces champs.

Composants de texte Les composants TextArea ou TextInput permettent d'afficher ou de récupérer du texte dans vos applications. Le composant TextArea est semblable à un champ de texte dynamique avec barres de défilement intégrées. Le composant TextInput est similaire à un champ de texte de saisie. Les deux composants présentent des fonctionnalités supplémentaires par rapport aux champs de texte équivalents ; toutefois, ils augmentent la taille des fichiers dans votre application. .

REMARQUE

Tous les champs de texte prennent en charge le codage Unicode. Pour plus d'informations sur l'Unicode, consultez la section « Présentation des chaînes et de la classe String », à la page 443.

Les méthodes de la classe TextField vous permettent de définir, sélectionner et manipuler du texte dans un champ de texte dynamique ou de saisie que vous créez en cours de programmation ou à l'exécution. Pour plus d'informations, voir « Utilisation de la classe TextField », à la page 370. Pour plus d'informations sur les champs de texte de débogage lors de l'exécution, consultez le guide *Utilisation de Flash*.

ActionScript propose également différentes manières de formater vos textes à l'exécution. La classe TextFormat vous permet de définir le formatage des caractères et des paragraphes pour les objets TextField (consultez « Utilisation de la classe TextFormat », à la page 410). Flash Player prend également en charge un sous-ensemble de balises HTML à utiliser pour mettre le texte en forme (consultez « Utilisation de texte au format HTML », à la page 427). Flash Player 7 et les versions ultérieures prennent en charge la balise HTML `img`, qui permet non seulement d'intégrer des images externes, mais également des fichiers SWF externes, ainsi que les clips qui résident dans la bibliothèque (consultez « Balise Image », à la page 431).

A partir de Flash Player7, vous pouvez appliquer des styles CSS (feuilles de style en cascade) aux champs de texte à l'aide de la classe TextField.StyleSheet. Vous pouvez utiliser le style CSS pour l'appliquer aux balises HTML intégrées, définir de nouvelles balises de format ou appliquer des styles. Pour plus d'informations sur l'utilisation de CSS, consultez « Formatage de texte avec les feuilles de style CSS », à la page 413.

Vous pouvez également directement affecter du texte au format HTML, pouvant éventuellement utiliser des styles CSS, à un champ de texte. Dans Flash Player 7 et les versions ultérieures, le texte HTML que vous assignez à un champ de texte peut contenir des supports intégrés (clips vidéo, fichiers SWF et fichiers JPEG). Flash Player 8 et les versions ultérieures permettent également de charger dynamiquement des images PNG, GIF et JPEG *progressives* (Flash Player 7 ne prend pas en charge les images JPEG progressives). Le texte entoure le média intégré comme le fait un navigateur Web dans un document HTML. Pour plus d'informations, voir « Balise Image », à la page 431.

Pour plus d'informations sur la terminologie comparant le texte, les chaînes, etc., reportez-vous à l'introduction de ce chapitre, « Utilisation du texte et des chaînes », à la page 367.

Utilisation de la classe TextField

La classe TextField représente les champs de texte dynamiques ou de saisie (modifiables) que vous créez à l'aide de l'outil Texte dans Flash. Utilisez les méthodes et les propriétés de cette classe pour contrôler les champs de texte à l'exécution. Les objets TextField prennent en charge les mêmes propriétés que les objets MovieClip à l'exception des propriétés `_currentframe`, `_droptarget`, `_framesloaded` et `_totalframes`. Vous pouvez obtenir et définir des propriétés et invoquer des méthodes pour les champs de texte de façon dynamique.

Pour contrôler un champ de texte dynamique ou de saisie en utilisant ActionScript, vous devez affecter au champ de texte un nom d'occurrence dans l'inspecteur des propriétés. Vous pouvez ensuite faire référence au champ de texte avec le nom de l'occurrence et utiliser les méthodes et les propriétés de la classe TextField pour contrôler le contenu ou l'aspect général du champ de texte.

Vous pouvez également créer des objets TextField à l'exécution et leur affecter des noms d'occurrence à l'aide de la méthode `MovieClip.createTextField()`. Pour plus d'informations, voir « Création de champs de texte à l'exécution », à la page 374.

Pour plus d'informations sur l'utilisation de la classe TextField, consultez les rubriques suivantes :

- « Affectation de texte à un champ de texte à l'exécution », à la page 371
- « A propos des noms d'occurrence et de variable de champ de texte », à la page 373

Pour des exemples qui décrivent comment utiliser les champs de texte à l'aide du code ActionScript, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/TextFields afin d'accéder à ces exemples.

- textfieldsA.fla
- textfieldsB.fla

Affectation de texte à un champ de texte à l'exécution

Lorsque vous créez des applications à l'aide de Flash, vous pouvez souhaiter charger du texte provenant d'une source externe, telle qu'un fichier texte, un fichier XML ou même un service Web distant. Flash vous permet de contrôler parfaitement la création et l'affichage de texte sur scène, par exemple le texte de support au format HTML, texte brut ou XML ainsi que les feuilles de style externes. Vous pouvez également utiliser ActionScript pour définir une feuille de style.

Pour affecter du texte à un champ de texte, effectuez l'une des opérations suivantes : utilisez les propriétés du texte ou de htmlTEXT ; créez un nom de variable pour le champ de texte dans le champ Var : dans l'inspecteur des propriétés et affectez-lui du texte ; affectez une valeur en liant le champ de texte à un champ de texte dans un autre composant.

L'exercice suivant permet d'affecter du texte à un champ de texte à l'exécution :

Pour affecter du texte à un champ de texte à l'exécution :

1. En utilisant l'outil Texte, créez un champ de texte sur la scène.
2. Après avoir sélectionné le champ de texte, dans l'inspecteur des propriétés (Fenêtre > Propriétés > Propriétés), sélectionnez Texte de saisie dans le menu contextuel Type de texte, puis saisissez **headline_txt** dans le champ Nom de l'Occurrence.

   Les noms d'occurrence peuvent uniquement comporter des lettres, des chiffres, des traits de soulignement (_) et des dollars ($).
3. Sélectionnez l'image 1 dans le scénario et ouvrez le panneau Actions (Fenêtre > Actions).

4. Tapez le code suivant dans le panneau Actions :

```
headline_txt.text = "New articles available on Developer Center";
```

5. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

Vous pouvez également créer un champ de texte à l'aide d'ActionScript, puis lui affecter du texte. Saisissez le code ActionScript suivant sur l'image 1 du scénario :

```
this.createTextField("headline_txt", this.getNextHighestDepth(), 100, 100,
  300, 20);
headline_txt.text = "New articles available on Developer Center";
```

Ce code crée un nouveau champ de texte portant le nom d'occurrence `headline_txt`. Le champ de texte est créé à la profondeur maximale suivante, aux coordonnées *x* et *y* de 100, 100, avec une largeur de champ de texte de 200 pixels et une hauteur de 20 pixels. Lorsque vous testez le fichier SWF (Contrôle > Tester l'animation), le texte « Nouveaux articles disponibles sur Developer Center » apparaît sur la scène.

Pour créer un champ de texte au format HTML, procédez comme suit :

Utilisez l'une des deux étapes suivantes pour activer le formatage HTML pour le champ de texte :

- Sélectionnez un champ de texte et cliquez sur le bouton Rendre le texte au format HTML dans l'inspecteur des propriétés.
- Définissez la propriété de champ de texte `html` sur `true` à l'aide d'ActionScript (voir l'exemple de code suivant).

Pour appliquer le formatage HTML à un champ de texte à l'aide d'ActionScript, saisissez le code ActionScript suivant dans l'Image 1 du scénario :

```
this.createTextField("headline_txt", this.getNextHighestDepth(), 100, 100,
  300, 20);
headline_txt.html = true;
headline_txt.htmlText = "New articles available on <i>Developer
  Center</i>.";
```

Le code précédent crée de manière dynamique un champ de texte, active la mise en forme HTML et affiche le texte « Nouveaux articles disponibles sur Developer Center » sur la scène, le terme « Developer Center » apparaissant en italiques.

ATTENTION

Lorsque vous utilisez du texte formaté HTML avec un champ de texte (pas des composants) sur la scène, vous devez affecter le texte à la propriété `htmlText` du champ de texte et non de la propriété de texte.

Pour des exemples qui décrivent comment utiliser les champs de texte à l'aide du code ActionScript, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/TextFields afin d'accéder à ces exemples.

- textfieldsA.fla
- textfieldsB.fla

A propos des noms d'occurrence et de variable de champ de texte

Dans le champ Nom de l'occurrence de l'inspecteur des propriétés, vous devez affecter un nom d'occurrence à un champ de texte pour invoquer des méthodes, puis obtenir et définir des propriétés dans ce champ de texte.

Dans la zone de texte Var de l'inspecteur de propriétés, attribuez un nom de variable à un champ texte dynamique ou de saisie. Vous pouvez alors attribuer des valeurs à la variable. Il s'agit d'une fonctionnalité déconseillée, que vous pouvez néanmoins utiliser lors de la création d'applications pour des versions antérieures de Flash Player (telles que Flash Player 4). Lorsque vous ciblez des lecteurs plus récents, ciblez le texte d'un champ de texte en utilisant son nom d'occurrence et ActionScript.

Veillez cependant à ne pas confondre le nom d'occurrence d'un champ texte avec son nom de variable. Le nom de variable d'un champ de texte est une référence de variable au texte contenu dans ce champ de texte, il ne s'agit pas d'une référence à un objet.

Par exemple, si vous affectez à un champ de texte le nom de variable `myTextVar`, vous pouvez ensuite utiliser le code suivant pour définir le contenu du champ de texte :

```
var myTextVar:String = "This is what will appear in the text field";
```

Toutefois, vous ne pouvez pas utiliser le nom de variable `myTextVar` pour définir la propriété `text` du champ de texte. Vous devez utiliser le nom d'occurrence, comme indiqué dans le code suivant :

```
// Cela ne fonctionnera pas.
myTextVar.text = "A text field variable is not an object reference";

// Pour un champ de texte de saisie avec le nom d'occurrence « myField »,
// cela fonctionnera.
myField.text = "This sets the text property of the myField object";
```

Utilisez la propriété `TextField.text` pour contrôler le contenu d'un champ de texte, sauf si vous ciblez une version de Flash Player qui ne prend pas en charge la classe TextField. Ceci réduit l'éventualité d'un conflit de noms de variables, qui pourrait engendrer un comportement inattendu à l'exécution.

Pour des exemples qui décrivent comment utiliser les champs de texte à l'aide du code ActionScript, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/TextFields afin d'accéder à ces exemples.

- textfieldsA.fla
- textfieldsB.fla

Création de champs de texte à l'exécution

Vous pouvez utiliser la méthode `createTextField()` de la classe MovieClip pour créer un champ de texte vide sur la scène à l'exécution. Ce nouveau champ est associé au scénario du clip qui appelle la méthode.

Pour créer dynamiquement un champ de texte à l'aide d'ActionScript :

1. Sélectionnez Fichier > Nouveau, puis sélectionnez Document Flash pour créer un nouveau fichier FLA.
2. Saisissez le code ActionScript suivant sur l'image 1 du scénario :

   ```
   this.createTextField("test_txt", 10, 0, 0, 300, 100);
   ```

   Ce code crée un champ de texte de 300 x 100 pixels appelé `test_txt` à l'emplacement (0, 0) et d'une profondeur (*ordre*-z) de 10.
3. Pour accéder aux méthodes et propriétés du nouveau champ de texte, utilisez le nom d'occurrence spécifié dans le premier paramètre de la méthode `createTextField()`.

   Par exemple, le code suivant crée un nouveau champ de texte nommé `test_txt`, puis modifie ses propriétés pour en faire un champ de texte multiligne avec retour automatique à la ligne, qui se développe pour s'ajuster à la taille du texte inséré. Il affecte ensuite du texte à la propriété `text` du champ de texte :

   ```
   test_txt.multiline = true;
   test_txt.wordWrap = true;
   test_txt.autoSize = "left";
   test_txt.text = "Create new text fields with the
     MovieClip.createTextField() method.";
   ```
4. Choisissez Contrôle > Tester l'animation pour visualiser le champ de texte.

   Le texte est créé au moment de l'exécution et apparaît sur la scène.

Vous pouvez utiliser la méthode `TextField.removeTextField()` pour supprimer un champ de texte créé avec `createTextField()`. La méthode `removeTextField()` ne fonctionne pas pour les champs de texte placés par le scénario au cours de la programmation.

Pour plus d'informations, consultez les sections createTextField (méthode MovieClip.createTextField) et removeTextField (méthode TextField.removeTextField) dans le *Guide de référence du langage ActionScript 2.0*.

REMARQUE

Certaines propriétés TextField telles que `_rotation` ne sont pas disponibles lorsque vous créez des champs de texte à l'exécution. Vous ne pouvez faire pivoter un champ de texte que s'il utilise des polices incorporées. Voir « Pour intégrer un symbole de police : », à la page 387.

Pour des exemples qui décrivent comment utiliser les champs de texte à l'aide du code ActionScript, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/TextFields afin d'accéder à ces exemples.

- textfieldsA.fla
- textfieldsB.fla

A propos de la manipulation des champs de texte

Vous pouvez manipuler de différentes manières les champs de texte que vous créez dans un fichier FLA. Vous pouvez manipuler un champ de texte à condition de lui affecter un nom d'occurrence dans l'inspecteur des propriétés, ou en affecter un à l'aide d'un code si vous utilisez du code pour créer le champ. L'exemple suivant est simple, il crée un champ de texte, lui affecte du texte, puis modifie sa propriété de bordure :

```
this.createTextField("pigeon_txt", this.getNextHighestDepth(), 100, 100,
  200, 20);
pigeon_txt.text = "I like seeds";
pigeon_txt.border = true;
```

Pour obtenir une liste complète des propriétés de la classe TextField, consultez le *Guide de référence du langage ActionScript 2.0* .

Pour examiner des exemples de manipulation de champs de texte, consultez les sections suivantes :

- « Modification de la position d'un champ de texte », à la page 376
- « Modification des dimensions d'un champ de texte à l'exécution », à la page 376

Pour des exemples qui décrivent comment utiliser les champs de texte à l'aide du code ActionScript, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/TextFields afin d'accéder à ces exemples.

- textfieldsA.fla
- textfieldsB.fla

Modification de la position d'un champ de texte

Vous pouvez modifier la position d'un champ de texte sur la scène au moment de l'exécution. Pour ce faire, vous devez définir les nouvelles valeurs des propriétés _x et _y du champ de texte, comme dans l'exemple suivant.

Pour repositionner un champ de texte à l'aide d'ActionScript, procédez comme suit :

1. Créez un fichier FLA, puis enregistrez-le sous le nom de **positionText.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
this.createTextField("my_txt", 10, 0, 0, 300, 200);
my_txt.border = true;
my_txt.text = "Hello world";
my_txt._x = (Stage.width - my_txt._width) / 2;
my_txt._y = (Stage.height - my_txt._height) / 2;
```

3. Enregistrez le document Flash et sélectionnez Contrôle > Tester l'animation pour voir le champ de texte centré sur la scène.

Pour des exemples qui décrivent comment utiliser les champs de texte à l'aide du code ActionScript, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/TextFields afin d'accéder à ces exemples.

- textfieldsA.fla
- textfieldsB.fla

Modification des dimensions d'un champ de texte à l'exécution

Vous devrez peut-être obtenir ou définir les dimensions d'un champ de texte de manière dynamique à l'exécution, au lieu de le faire dans l'environnement auteur. L'exemple suivant crée un champ de texte dans un scénario et définir ses dimensions initiales sur 100 pixels de large par 21 pixels de haut. Par la suite, le champ de texte est redimensionné à 300 pixels de large sur 200 pixels de haut, et repositionné au centre de la scène.

Pour redimensionner un champ de texte à l'aide d'ActionScript :

1. Créez un document Flash, puis enregistrez-le sous le nom de **resizeText.fla**.

2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
this.createTextField("my_txt", 10, 0, 0, 100, 21);
my_txt.border = true;
my_txt.multiline = true;
my_txt.text = "Hello world";
my_txt.wordWrap = true;
my_txt._width = 300;
my_txt._height = 200;
my_txt._x = (Stage.width - my_txt._width) / 2;
my_txt._y = (Stage.height - my_txt._height) / 2;
```

3. Enregistrez le document Flash et sélectionnez Contrôle > Tester l'animation pour visualiser les résultats dans l'environnement auteur.

L'exemple précédent a permis de redimensionner un champ de texte créé de manière dymanique à 300 pixels sur 200 pixels à l'exécution, mais lorsque vous chargez un contenu à partir d'un site Web externe et que vous n'etes pas certain de la quantité de contenu qui sera retournée, cette technique n'est peut-être pas adaptée à vos besoins. Heureusement, Flash comprend une propriété `TextField.autoSize` que vous pouvez utiliser pour redimensionner automatiquement un champ de texte en fonction de son contenu. L'exemple suivant démontre la manière dont vous pouvez utiliser la propriété `TextField.autoSize` pour redimensionner le champ de texte après y avoir ajouté du texte.

Pour redimensionner automatiquement des champs de texte en fonction du contenu :

1. Créez un document Flash, puis enregistrez-le sous le nom de **resizeTextAuto.fla**.
2. Ajoutez le code suivant à l'image 1 du scénario principal :

```
this.createTextField("my_txt", 10, 10, 10, 160, 120);
my_txt.autoSize = "left";
my_txt.border = true;
my_txt.multiline = true;
my_txt.text = "Lorem ipsum dolor sit amet, consectetur adipisicing elit,
  sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
  Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
  nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in
  reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
  pariatur. Excepteur sint occaecat cupidatat non proident, sunt in
  culpa qui officia deserunt mollit anim id est laborum.";
my_txt.wordWrap = true;
```

REMARQUE

Si vous collez ce code directement dans le panneau Actions à partir de certaines versions de l'aide de Flash, vous pouvez rencontrer des sauts de ligne dans la chaîne de texte longue. Dans ce cas, le code ne sera pas compilé. Si vous vous trouvez dans cette situation, activez les Caractères masqués dans le menu contextuel du panneau Actions, puis supprimez les caractères de saut de ligne dans la chaîne de texte longue.

3. Enregistrez le document Flash et sélectionnez Contrôle > Tester l'animation pour visualiser le document Flash dans l'environnement auteur.

   Flash redimensionne le champ de texte verticalement de manière à afficher l'ensemble du contenu sans le réduire aux limites du champ. Si vous définissez la propriété `my_txt.wordWrap` sur `false`, le champ de texte est redimensionné horizontalement en fonction du texte.

   Pour appliquer une hauteur maximale au champ de texte redimensionné automatiquement (afin que la hauteur du champ de texte ne dépasse pas les limites de la scène), utilisez le code suivant.

```
if (my_txt._height > 160) {
  my_txt.autoSize = "none";
  my_txt._height = 160;
}
```

Vous devez ajouter une fonctionnalité de défilement, par exemple une barre de défilement, afin de permettre aux utilisateurs de visualiser le reste du texte. Vous pouvez aussi faire passer le pointeur de la souris sur le texte ; cette méthode est souvent adéquate pour tester ce code.

Pour des exemples qui décrivent comment utiliser les champs de texte à l'aide du code ActionScript, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/TextFields afin d'accéder à ces exemples.

- textfieldsA.fla
- textfieldsB.fla

A propos du chargement de texte et de variables dans des champs de texte

Vous pouvez charger du texte dans un document Flash de différentes manières, y compris (mais pas seulement) à l'aide de FlashVars, LoadVars, XML ou de services Web. La méthode la plus simple pour intégrer du texte dans un document Flash consiste peut-être à utiliser la propriété FlashVars, qui fait passer des chaînes de texte courtes dans un document Flash au moyen des balises `object` et `embed` dans le code HTML que vous utilisez pour incorporer le fichier SWF dans une page HTML. Un autre moyen facile de charger du texte ou des variables dans un document Flash consiste à utiliser la classe LoadVars, qui permet de charger de gros blocs de texte ou une série de variables porteuses d'URL à partir d'un fichier texte.

Comme vous l'avez vu dans les exemples précédents de cette section, certaines méthodes de chargement de texte dans un fichier SWF sont plus simples que d'autres. Toutefois, si vous diffusez des données provenant de sites externes, vous risquez de ne pas avoir le choix quant au format des données que vous devez charger.

Chaque manière de charger et/ou d'envoyer des données vers et en provenance d'un fichier SWF possède ses avantages et ses inconvénients. Le XML, les services Web et le Flash Remoting sont les outils les plus polyvalents pour charger des données externes, mais ils sont également les plus difficiles à apprendre. Pour plus d'informations sur le Flash Remoting, consultez le site www.adobe.com/support/flashremoting.

Les solutions FlashVars et LoadVars sont beaucoup plus simples, comme vous pouvez le constater aux rubriques « Utilisation de FlashVars pour charger et afficher du texte », à la page 380 et « Utilisation de LoadVars pour charger et afficher du texte », à la page 381, mais ces méthodes peuvent être plus limitées en termes de types et de formats des données que vous pouvez charger. Vous devez également respecter les restrictions de sécurité lorsque vous envoyez et chargez des données. Pour plus d'informations sur la sécurité, consultez le Chapitre 16, « Fonctionnement de la sécurité » Pour plus d'informations sur le chargement de données externes, consultez le Chapitre 15, « Utilisation de données externes »

Les sections suivantes présentent divers moyens de charger du texte et des variables dans vos documents :

Pour des exemples qui décrivent comment utiliser les champs de texte à l'aide du code ActionScript, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/LoadText afin d'accéder à ces exemples.

- loadText.fla
- formattedText.fla

Pour un exemple de fichier source, aliasing.fla, qui charge du texte et applique la mise en forme anti-aliasée en plus de la mise en cache bitmap, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Advanced Anti-Aliasing afin d'accéder à ces exemples.

Utilisation de FlashVars pour charger et afficher du texte

L'utilisation de FlashVars est simple, mais requiert la publication de vos fichiers SWF en plus des documents HTML. Vous modifiez le code HTML généré et incluez les propriétés FlashVars dans les balises `object` et `embed`. Vous pouvez tester le document Flash en visualisant le document HTML modifié dans votre navigateur Web.

Pour utiliser FlashVars pour transmettre des variables de HTML vers votre document Flash :

1. Créez un document Flash, puis enregistrez-le sous le nom de **flashvars.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
this.createTextField("my_txt", 10, 10, 10, 100, 21);
my_txt.text = _level0.username;
```

3. Enregistrez le document Flash et sélectionnez Fichier > Publier pour générer les fichiers HTML et SWF.

REMARQUE	Par défaut, un document HTML est publié dans le même répertoire que votre fichier FLA. Si un document HTML n'est pas publié, sélectionnez Fichier › Paramètres de publication, puis choisissez l'onglet Formats. Veillez à sélectionner l'option HTML.

4. Ouvrez le document flashvars.html dans un éditeur de texte ou HTML.

5. Dans le document HTML, modifiez le code dans la balise `object` selon ce qui suit.

   Le code que vous devez ajouter apparaît en **gras**.

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
  codebase="http://fpdownload.adobe.com/pub/shockwave/cabs/flash/
  swflash.cab#version=8,0,0,0" width="550" height="400" id="flashvars"
  align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="flashvars.swf" />
<param name="FlashVars" value="username=Thomas" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="flashvars.swf" FlashVars="username=Thomas" quality="high"
  bgcolor="#ffffff" width="550" height="400" name="flashvars"
  align="middle" allowScriptAccess="sameDomain" type="application/x-
  shockwave-flash" pluginspage="http://www.adobe.com/go/getflashplayer"
  />
</object>
```

6. Enregistrez vos modifications dans le document HTML.
7. Ouvrez le document HTML modifié dans un navigateur Web.

   Le fichier SWF affiche le nom « Thomas » dans le champ de texte créé de manière dynamique sur la scène.

Pour plus d'informations sur la sécurité, voir Chapitre 16, « Fonctionnement de la sécurité »

Utilisation de LoadVars pour charger et afficher du texte

Vous pouvez également utiliser la classe LoadVars pour charger du contenu dans un fichier SWF qui charge du texte ou des variables depuis un fichier externe du même serveur ou encore du contenu provenant d'un autre serveur. L'exemple suivant décrit la procédure à suivre pour créer un champ de texte de manière dynamique et le remplir à l'aide du contenu d'un fichier texte distant.

Pour utiliser LoadVars pour remplir un champ de texte avec du texte externe :

1. Créez un document Flash, puis enregistrez-le sous le nom de **loadvarsText.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
this.createTextField("my_txt", 10, 10, 10, 320, 100);
my_txt.autoSize = "left";
my_txt.border = true;
my_txt.multiline = true;
my_txt.wordWrap = true;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function (src:String):Void {
  if (src != undefined) {
    my_txt.text = src;
  } else {
    my_txt.text = "Unable to load external file.";
  }
}
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

   Le premier bloc de code de la section précédente crée un nouveau champ de texte sur la scène et active l'affichage multiligne et le retour à la ligne. Le deuxième bloc de code définit un nouvel objet LoadVars qui est utilisé pour charger un fichier texte (lorem.txt) à partir d'un serveur Web distant et afficher son contenu dans le champ de texte my_txt créé antérieurement.

3. Enregistrez le document Flash, puis choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

   Après un court délai, Flash affiche le contenu du fichier distant dans le champ de texte sur la scène.

Pour plus d'informations sur la sécurité, voir Chapitre 16, « Fonctionnement de la sécurité »

Chargement de variables à l'aide de LoadVars

La classe LoadVars vous permet également de charger des variables au format encodé par URL, une opération similaire au transfert de variables dans la chaîne de requête d'un navigateur Web. L'exemple suivant présente le chargement d'un fichier texte distant dans un fichier SWF et l'affichage de ses variables, `monthNames` et `dayNames`.

Pour charger des variables à partir d'un fichier texte à l'aide de LoadVars :

1. Créez un document Flash, puis enregistrez-le sous le nom de **loadvarsVariables.fla**.
2. Ajoutez le code suivant à l'image 1 du scénario :

```
this.createTextField("my_txt", 10, 10, 10, 320, 100);
my_txt.autoSize = "left";
my_txt.border = true;
my_txt.multiline = true;
my_txt.wordWrap = true;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onLoad = function (success:Boolean):Void {
  if (success) {
    my_txt.text = "dayNames: " + lorem_lv.dayNames + "\n\n";
    my_txt.text += "monthNames: " + lorem_lv.monthNames;
  } else {
    my_txt.text = "Unable to load external file.";
  }
}
/* contenu de params.txt :
  &monthNames=January,February,...&dayNames=Sunday,Monday,...
*/
lorem_lv.load("http://www.helpexamples.com/flash/params.txt");
```

3. Enregistrez le document Flash et sélectionnez Contrôle > Tester l'animation dans le menu principal.

   Comme vous utilisez la méthode `LoadVars.onLoad()` au lieu de `LoadVars.onData()`, Flash analyse les variables et crée des variables dans l'occurrence d'objet LoadVars. Le fichier texte externe contient deux variables, `monthNames` et `dayNames`, qui contiennent toutes les deux des chaînes de caractères.

Pour plus d'informations sur la sécurité, voir Chapitre 16, « Fonctionnement de la sécurité »

Chargement et affichage de texte à partir d'un document XML

XML est une technique très couramment employée pour distribuer du contenu sur Internet, en partie parce que ce standard est largement accepté pour l'organisation et l'analyse des données. En tant que tel, le XML constitue un excellent choix pour envoyer et recevoir des données de Flash ; toutefois, le XML est légèrement plus difficile à apprendre que l'utilisation de LoadVars et FlashVars pour charger des données et afficher du texte.

Pour charger du texte dans Flash depuis un document XML externe :

1. Créez un document Flash, puis enregistrez-le sous le nom de **xmlReviews.fla**.
2. Ajoutez le code suivant à l'image 1 du scénario :

```
this.createTextField("my_txt", 10, 10, 10, 320, 100);
my_txt.autoSize = "left";
my_txt.border = true;
my_txt.multiline = true;
my_txt.wordWrap = true;

var reviews_xml:XML = new XML();
reviews_xml.ignoreWhite = true;
reviews_xml.onLoad = function (success:Boolean):Void {
  if (success) {
    var childItems:Array = reviews_xml.firstChild.childNodes;
    for (var i:Number = 0; i < childItems.length; i++) {
      my_txt.text += childItems[i].firstChild.firstChild.nodeValue +
"\n";
    }
  } else {
    my_txt.text = "Unable to load external file.";
  }
}
reviews_xml.load("http://www.helpexamples.com/flash/xml/reviews.xml");
```

   Le premier bloc de code de la section précédente crée un nouveau champ de texte sur la scène. Ce champ de texte est utilisé pour afficher différentes parties du document XML qui est chargé par la suite. Le deuxième bloc de code gère la création d'un objet XML qui sera utilisé pour charger le contenu XML. Une fois les données entièrement chargées et analysées par Flash, le gestionnaire d'événement `XML.onLoad()` est invoqué et affiche le contenu du paquet XML dans le champ de texte.

3. Enregistrez le document Flash, puis choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

   Flash affiche le résultat suivant dans le champ de texte sur la scène :

```
Item 1
Item 2
...
Item 8
```

Pour plus d'informations sur la sécurité, voir Chapitre 16, « Fonctionnement de la sécurité »

Utilisation des polices

Les polices sont des jeux de caractères présentant un aspect, un style et une taille similaires. Quels que soient les objets que vous créez à l'aide de Flash, vous utiliserez probablement du texte avec au moins une ou deux polices dans vos applications Flash. Si vous créez des animations et que vous ne savez pas au juste si vos utilisateurs finaux disposent d'une police spécifique sur leur système, vous devez comprendre les éléments de base de l'incorporation de polices.

Les sections suivantes présentent l'intégration de caractères, de polices complètes, de polices partagées et d'autres techniques d'utilisation des polices dans Flash.

Pour plus d'informations sur les polices, consultez les sections suivantes :

- « Intégration de caractères », à la page 386
- « Intégration des polices », à la page 387
- « Création de jeux de caractères personnalisés », à la page 390
- « Utilisation des méthodes TextField avec les polices intégrées », à la page 392
- « A propos du partage des polices », à la page 394

L'exemple suivant présente la manière d'ajouter et de supprimer des caractères et des jeux de caractères incorporés dans un document Flash.

Pour ajouter et supprimer des caractères et des jeux de caractères incorporés :

1. Créez un document Flash, puis enregistrez-le sous le nom de **embedding.fla**.
2. Créez un champ de texte dynamique sur la scène à l'aide de l'outil Text.
3. Cliquez sur Intégrer pour ouvrir la boîte de dialogue Caractères incorporés.
4. Sélectionnez un jeu de caractères spécifique à intégrer en cliquant dessus à l'aide du pointeur de la souris.

   Pour sélectionner plusieurs jeux de caractères, vous pouvez utiliser la touche Maj ou Ctrl tout en sélectionnant des éléments à l'aide du pointeur de la souris. Pour sélectionner un bloc de jeux de caractères, sélectionnez un jeu à l'aide du pointeur de la souris, appuyez sur la touche Maj et maintenez-la enfoncée, puis cliquez sur un nouveau jeu de caractères. L'utilisation de la touche Maj sélectionne tous les jeux de caractères situés entre les deux jeux sélectionnés. Pour sélectionner plusieurs jeux de caractères non consécutifs, appuyez sur la touche Ctrl et maintenez-la enfoncée pendant que vous sélectionnez les jeux de caractères. Vous pouvez également sélectionner rapidement plusieurs jeux de caractères en sélectionnant le premier à l'aide de la souris et, tout en maintenant le bouton de la souris enfoncé, en faisant glisser celle-ci sur plusieurs jeux de caractères.

5. Pour supprimer un jeu de caractères spécifique que vous avez ajouté antérieurement, appuyez sur la touche Ctrl et maintenez-la enfoncée, puis désélectionnez le jeu de caractères en cliquant dessus à l'aide du pointeur de la souris.
6. Pour supprimer tous les jeux de caractères et les éventuels caractères spécifiés dans le champ de saisie de texte Inclure ces caractères, cliquez sur Ne pas intégrer.

   L'option Ne pas intégrer efface les éventuels caractères individuels ou jeux de caractères sélectionnés précédemment.

ATTENTION	Si vous cliquez sur Ne pas intégrer dans la boîte de dialogue Caractères incorporés, vous supprimez les éventuels caractères et jeux de caractères intégrés spécifiés qui ont été sélectionnés précédemment sans vous demander de confirmation.

Intégration de caractères

Si vous travaillez avec des polices intégrées et savez exactement de quels caractères vous avez besoin, vous pouvez réduire la taille du fichier en intégrant uniquement les caractères dont vous avez besoin au lieu d'ajouter d'autres contours de polices qui ne vous seront pas utiles. Pour intégrer certains caractères dans un champ de texte, mais ne pas intégrer un jeu complet de caractères, utilisez la boîte de dialogue Caractères incorporés pour spécifier les caractères que vous souhaitez intégrer.

Pour intégrer des caractères spécifiques à utiliser dans un champ de texte :

1. Créez un document Flash, puis enregistrez-le sous le nom de **charembed.fla**.
2. A l'aide du toucher Text, créez un champ de texte sur la scène et définissez le type de texte du champ de texte comme dynamique ou saisie.
3. Alors que le champ de texte est toujours sélectionné sur la scène, cliquez sur Intégrer dans l'inspecteur des propriétés afin d'ouvrir la boîte de dialogue Caractères incorporés.

   La boîte de dialogue Caractères incorporés vous permet de définir les jeux de caractères qui seront intégrés dans le document Flash (ainsi que le nombre de glyphes par jeu de caractères), de spécifier quels caractères intégrer et vous indique le nombre total de glyphes qui sont intégrés dans ce champ de texte.
4. Saisissez la chaîne **hello world** dans le champ Inclure ces caractères.

   La boîte de dialogue vous indique que 8 glyphes au total seront incorporés pour ce champ de texte. Bien que la chaîne « hello world » contienne 11 caractères, Flash intègre seulement les glyphes uniques, de sorte que les lettres l et o sont intégrés une seule fois.
5. Cliquez sur OK pour appliquer les modifications et revenir à votre document.

6. En utilisant l'outil Texte, créez un champ de texte sur la scène.
7. Définissez le type de texte de ce champ sur dynamique dans l'inspecteur des propriétés.
8. Saisissez la chaîne **hello world** dans le champ de texte sur la scène.
9. Cliquez sur Intégrer dans l'inspecteur des propriétés pour ouvrir de nouveau la boîte de dialogue Caractères incorporés.
10. Cliquez sur Remplissage automatique pour remplir automatiquement le champ Inclure ces caractères.

    Vous verrez apparaître la chaîne « helo wrd ». Au lieu d'avoir à indiquer à Flash quels caractères vous souhaitez inclure, Flash peut déterminer pour vous tous les caractères uniques du champ de texte spécifié.

CONSEIL	Flash peut déterminer les caractères à intégrer automatiquement uniquement si le champ de texte contient du texte sur la scène. Si le champ de texte est rempli à l'aide du code ActionScript, vous devez spécifier quels caractères vous souhaitez intégrer dans ce champ.

11. Cliquez sur OK.

Intégration des polices

Lorsque vous intégrez des polices, Flash stocke l'ensemble des informations relatives à la police dans le fichier SWF, de sorte que la police s'affiche correctement, même si elle n'est pas installée sur l'ordinateur de l'utilisateur. Si vous utilisez dans votre fichier FLA une police qui est absente sur le système de l'utilisateur sans l'intégrer dans le fichier SWF, Flash Player sélectionnera automatiquement une police de remplacement.

REMARQUE	Vous devez intégrer une police uniquement si vous utilisez des champs de texte dynamiques ou de saisie. Si vous utilisez un champ de texte statique, vous n'avez pas à intégrer la police.

Pour intégrer un symbole de police :

1. Choisissez Fenêtre > Bibliothèque pour ouvrir la bibliothèque du fichier FLA.

   Ouvrez la bibliothèque à laquelle vous souhaitez ajouter le symbole de police.
2. Sélectionnez Nouvelle police dans le menu contextuel de la bibliothèque (coin supérieur droit du panneau Bibliothèque).
3. Dans la boîte de dialogue Propriétés des symboles de police, saisissez un nom pour le symbole de police dans le champ Nom.

4. Sélectionnez une police dans le menu Police ou saisissez un nom de police dans le champ Police.
5. Pour appliquer un style à la police, sélectionnez Gras, Italique ou Texte aliasé.
6. Saisissez la taille de police à intégrer, puis cliquez sur OK pour appliquer les modifications et revenir à votre document.

   Votre police apparaît maintenant dans la bibliothèque du document actif.

Après avoir intégré une police dans votre bibliothèque, vous pouvez l'utiliser avec un champ de texte sur la scène.

Pour utiliser un symbole de police intégré dans votre document Flash :

1. Suivez les étapes de la procédure décrite à la section « Intégration des polices », à la page 387 pour intégrer une police dans votre bibliothèque.
2. Utilisez l'outil Texte pour créer un champ de texte sur la scène.
3. saisissez du texte dans ce champ.
4. Sélectionnez le champ de texte et ouvrez l'inspecteur des propriétés.
   a. Définissez-le en tant que champ de texte à une seule ligne.
   b. Sélectionnez le nom de la police intégrée en utilisant le menu déroulant Police.

   Le nom des polices intégrées est suivi d'un astérisque (*).
5. Cliquez sur Intégrer dans l'inspecteur des propriétés pour ouvrir la boîte de dialogue Caractères incorporés.

   La boîte de dialogue Intégration de caractères vous permet de sélectionner les caractères individuels ou les jeux de caractères que vous souhaitez intégrer pour le champ de texte sélectionné. Pour spécifier quels caractères incorporer, saisissez les caractères dans le champ de la boîte de dialogue, ou cliquez sur Remplissage automatique pour remplir automatiquement le champ de texte à l'aide des caractères uniques figurant actuellement dans le champ de texte. Si vous n'êtes pas certain des caractères dont vous avez besoin (par exemple, si votre texte est chargé à partir d'un fichier externe ou d'un service Web), vous pouvez sélectionner des jeux de caractères entiers à intégrer, tels que les Majuscules [A..Z], les minuscules [a..z], les chiffres [0..9], les signes de ponctuation [!@#%...] et les jeux de caractères correspondant à différentes langues.

REMARQUE	Chaque jeu de caractères que vous sélectionnez augmente la taille finale du fichier SWF car Flash doit stocker toutes les informations relatives à la police pour chaque jeu de caractères que vous utilisez.

6. Sélectionnez les caractères individuels ou les jeux de caractères que vous souhaitez intégrer, puis cliquez sur OK pour appliquer les modifications et revenir à votre document.
7. Choisissez Contrôle > Tester l'animation pour tester le document Flash dans l'environnement de programmation.

   La police intégrée s'affiche dans le champ de texte sur la scène. Pour vérifier correctement que la police est intégrée, vous devrez peut-être effectuer un test sur un autre ordinateur ne disposant pas de cette police.

   Vous pouvez aussi définir les propriétés `TextField._alpha` ou `TextField._rotation` du champ de texte avec les polices intégrées, car ces propriétés ne fonctionnent sur sur les polices intégrées (voir les étapes suivantes).
8. Fermez le fichier SWF pour revenir dans l'environnement de programmation.
9. Sélectionnez le champ de texte sur la scène et ouvrez l'inspecteur des propriétés.
   a. Définissez le type du champ de texte sur Texte dynamique.
   b. Saisissez **font_txt** dans le champ Nom de l'occurrence.
10. Ajoutez le code suivant à l'image 1 du scénario :

```
font_txt._rotation = 45;
```

11. Choisissez de nouveau Contrôle > Tester l'animation pour visualiser les modifications dans l'environnement auteur.

    La police sélectionnée pivote de 45 ° dans le sens des aiguilles d'une montre et vous pouvez toujours voir le texte car il est intégré au fichier SWF.

ATTENTION Si vous n'intégrez pas une police dans votre document Flash et que Flash Player choisit automatiquement une police de substitution sur l'ordinateur de l'utilisateur, la propriété TextField.font utilise la police d'origine employée dans le fichier FLA, et non la police de substitution.

REMARQUE Si vous utilisez des polices intégrées avec différents styles dans vos champs de texte, vous devez intégrer le style à utiliser. Par exemple, si vous utilisez une police intégrée appelée Times, puis que vous souhaitez qu'un mot apparaisse en italiques, vous devez intégrer à la fois les contours de police normal et italiques. Dans le cas contraire, le texte n'apparaîtra pas dans le champ.

Création de jeux de caractères personnalisés

Outre l'utilisation des jeux de caractères par défaut de Flash, vous pouvez également créer vos propres jeux de caractères et les ajouter à la boîte de dialogue Caractères incorporés.
Par exemple, vous devrez peut-être permettre à certains champs d'intégrer la police Extended Latin pour prendre en charge différents caractères accentués. Toutefois, il est possible que vous n'ayez pas besoin des chiffres ni de la ponctuation, ou que vous ayez uniquement besoin de caractères en majuscules. Au lieu d'intégrer des jeux de caractères entiers, vous pouvez créer un jeu de caractères personnalisé contenant uniquement les caractères dont vous avez besoin. Cela vous permettra de limiter la taille de votre fichier SWF autant que possible, car vous ne stockerez pas d'informations supplémentaires sur les polices pour les caractères dont vous n'avez pas besoin.

Pour créer un jeu de caractères personnalisé, vous devez modifier le fichier UnicodeTable.xml situé dans le répertoire C:\Program Files\Adobe\Adobe Flash CS3\<*langue*>\First Run\FontEmbedding\. Ce fichier définit les jeux de caractères par défaut et les plages de caractères ainsi que les caractères qu'elles contiennent.

Avant de créer un jeu de caractères personnalisé, vous devez comprendre la structure XML nécessaire. Les nœuds XML suivants définissent le jeu de caractères Majuscules [A..Z] :

```
<glyphRange name="Uppercase [A..Z] " id="1" >
  <range min="0x0020" max ="0x0020" />
  <range min="0x0041" max ="0x005A" />
</glyphRange>
```

Vous pouvez remarquer que le noeud `glyphRange` comprend `name`, `Uppercase [A..Z]` et `id`. Un nœud `glyphRange` peut comporter autant de nœuds enfants *range* que nécessaire.
Une plage peut se composer d'un seul caractère, par exemple `0x0020` (le caractère d'espacement), figurant dans l'extrait précédent ou une série de caractères, telle que la deuxième plage constituant le nœud enfant. Pour intégrer un seul caractère, définissez les valeurs `min` et `max` sur la même valeur de caractère unicode.

Un autre exemple de nœud XML `glyphRange` est celui des chiffres `Numerals [0..9]` :

```
<glyphRange name="Numerals [0..9] " id="3" >
  <range min="0x0030" max ="0x0039" />
  <range min="0x002E" max ="0x002E" />
</glyphRange>
```

Cette plage de caractères comprend les valeurs Unicode 0x0030 (zéro) à 0x0039 (9), ainsi que la valeur 0x002E (.).

Avant de créer un jeu de caractères personnalisé, vous devez connaître les caractères et leurs valeurs Unicode correspondantes. Pour obtenir les valeurs Unicode, consultez le site Web des normes Unicode, www.unicode.org, qui contient le tableau des codes de caractères Unicode correspondant à des dizaines de langues.

ATTENTION Pour ajouter des jeux de caractères personnalisés, vous devez modifier un fichier XML dans le dossier d'installation de Flash. Avant de modifier ce fichier, vous devriez effectuer une copie de sauvegarde au cas où vous souhaiteriez revenir au tableau Unicode d'origine.

ATTENTION Adobe vous recommande de ne pas modifier les jeux de caractères existants qui sont installés à l'aide de Flash et de créer plutôt vos propres jeux de caractères personnalisés comprenant les caractères et les signes de ponctuation dont vous avez besoin.

Pour créer et utiliser un jeu de caractères personnalisé :

1. Ouvrez le document UnicodeTable.xml, situé dans le répertoire *<Répertoire d'installation de Flash>**<langue>*\First Run\FontEmbedding\ à l'aide d'un éditeur XML ou d'un éditeur de texte tel que Notepad ou TextEdit.

   REMARQUE N'oubliez pas de conserver une sauvegarde de ce document au cas où vous souhaiteriez revenir au fichier d'origine installé avec Flash.

2. Défilez jusqu'au bas du document XML et ajoutez le code XML suivant juste avant le nœud final `</fontEmbeddingTable>` :

```
<glyphRange name="Uppercase and Numerals [A..Z,0..9] " id="100" >
  <range min="0x0020" max ="0x0020" />
  <range min="0x002E" max ="0x002E" />
  <range min="0x0030" max ="0x0039" />
  <range min="0x0041" max ="0x005A" />
</glyphRange>
```

3. Enregistrez vos modifications dans UnicodeTable.xml.

   Si Flash est ouvert, vous devez redémarrer l'application avant d'utiliser le nouveau jeu de caractères.

4. Ouvrez ou redémarrez Flash, puis créez un document Flash.
5. Ajoutez une nouvelle occurrence de TextField sur la scène à l'aide de l'outil Text.

6. Définissez le type Text de TextField sur dynamique dans l'inspecteur des propriétés, puis cliquez sur Options de caractères incorporés pour ouvrir la boîte de dialogue Caractères incorporés.
7. Défilez jusqu'au bas de la boîte de dialogue Caractères incorporés et sélectionnez votre nouveau jeu de caractères personnalisé, Majuscules (Uppercase) et Chiffres (Numerals) [A..Z,0..9] (38 glyphes).
8. Sélectionnez les autres jeux de caractères éventuels, puis cliquez sur OK.

   Si vous sélectionnez votre jeu de caractères personnalisé, Uppercase et Numerals [A..Z,0..9], ainsi que le jeu de caractères par défaut Uppercase [A..Z] ou Numerals [0..9], remarquez que le nombre de glyphes incorporés ne change pas. C'est dû au fait que tous les caractères en majuscules sont inclus dans votre jeu de caractères personnalisé, et que Flash n'intègre pas de caractères e ndouble, ce qui permet de réduire autant que possible la taille du fichier. Si vous sélectionnez le jeu de caractères Ponctuation (Punctuation), qui comprend 52 glyphes, ainsi que votre jeu de caractères personnalisé, comprenant 38 glyphes, Flash stocke des informations relatives à 88 glyphes seulement au lieu de 90. En effet, deux caractères se recouvrent dans ces jeux : l'espacement et le point se trouvent déjà dans votre jeu de caractères personnalisé.

CONSEIL	La position d'un caractère définie dans la boîte de dialogue Caractères incorporés est déterminée par son emplacement dans le document XML. Vous pouvez modifier l'ordre des jeux de caractères, y compris de vos jeux de caractères personnalisés, en déplaçant les paquets ‹glyphRange› dans le fichier XML.

Utilisation des méthodes TextField avec les polices intégrées

Les méthodes de la classe TextField fournissent des fonctionnalités utiles à vos applications. Vous pouvez par exemple contrôler l'épaisseur d'un champ de texte à l'aide du code ActionScript, comme le décrit l'exemple suivant.

Pour définir l'épaisseur d'un champ de texte à l'aide d'ActionScript :

1. Créez un document Flash, puis enregistrez-le sous le nom de **textfieldThickness.fla**.
2. Ouvrez le panneau Bibliothèque et sélectionnez Nouvelle police dans le menu contextuel (dans le coin supérieur droit du panneau Bibliothèque).

   La boîte de dialogue Propriétés des symboles de police s'ouvre. Cette boîte de dialogue vous invite à sélectionner une police à intégrer dans le fichier SWF (ainsi qu'un style et une taille de police). Vous pouvez également affecter un nom de police affiché dans la bibliothèque du document et le menu déroulant Police de l'inspecteur des propriétés (si un champ de texte est sélectionné sur la scène).

   a. Sélectionnez le nom de police Times New Roman dans le menu déroulant Police.
   b. Assurez-vous de désactiver les options Gras et Italique.
   c. Définissez la taille sur 30 pixels.
   d. Saisissez le nom de police **Times (intégrée)**
   e. Cliquez sur OK.
3. Dans la bibliothèque, cliquez avec le bouton droit de la souris sur le symbole de la police et sélectionnez Liaison dans le menu contextuel.

   Flash affiche la boîte de dialogue Propriétés de liaison.
4. Sélectionnez Exporter pour ActionScript, puis Exporter dans la première image et cliquez sur OK.
5. Ajoutez l'ActionScript suivant à l'Image 1 du scénario :

```
// 1
this.createTextField("thickness_txt", 10, 0, 0, Stage.width, 22);
this.createTextField("lorem_txt", 20, 0, 20, Stage.width, 0);
lorem_txt.autoSize = "left";
lorem_txt.embedFonts = true;
lorem_txt.antiAliasType = "advanced";
lorem_txt.text = "Lorem ipsum dolor sit amet, consectetur adipisicing
  elit, sed do eiusmod tempor incididunt ut labore et dolore magna
  aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco
  laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
  in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
  pariatur. Excepteur sint occaecat cupidatat non proident, sunt in
  culpa qui officia deserunt mollit anim id est laborum.";
lorem_txt.wordWrap = true;

// 2
var style_fmt:TextFormat = new TextFormat();
style_fmt.font = "Times (embedded)";
style_fmt.size = 30;
lorem_txt.setTextFormat(style_fmt);
```

```
// 3
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  // Les valeurs de TextField.thickness peuvent varier de -200 à +200.
  lorem_txt.thickness = Math.round(_xmouse * (400 / Stage.width) - 200);
  thickness_txt.text = "TextField.thickness = " + lorem_txt.thickness;
};
Mouse.addListener(mouseListener);
```

Le premier bloc de code crée deux champs de texte, `thickness_txt` et `lorem_txt`, puis les place sur la scène. Le champ de texte `lorem_txt` définit sa propriété `embedFonts` sur `true` et remplit le champ de texte d'un bloc de texte.

Le deuxième bloc de code définit un format de texte avec la police Times New Roman, définit la taille de la police sur 30 pixels et applique le format de texte au champ `lorem_txt`.

Le troisième et dernier bloc de code définit un écouteur de souris et l'affecte à l'événement `onMouseMove`. Lorsque le pointeur de la souris se déplace horizontalement sur la scène, la propriété `TextField.thickness` varie de -200 à +200, selon la valeur actuelle de `_xmouse`.

6. Enregistrez vos modifications dans le fichier FLA.
7. Choisissez Contrôle > Tester l'animation pour tester votre document Flash.

   Lorsque vous déplacez le pointeur de la souris vers la moitié gauche de la scène, l'épaisseur de la police diminue. Lorsque vous déplacez le pointeur de la souris vers la moitié droite de la scène, l'épaisseur de la police augmente.

A propos du partage des polices

Pour utiliser une police en tant qu'élément de bibliothèque partagé, vous pouvez créer un symbole de police dans le panneau Bibliothèque, puis affecter les attributs suivants au symbole de police :

- Une chaîne d'identifiant
- Une URL sur laquelle le document contenant le symbole de police sera placé

Ainsi, vous pouvez effectuer une liaison avec la police et l'utiliser dans une application Flash sans que la police ne soit stockée dans un fichier FLA.

Présentation du rendu d'un texte anti-alias

Le rendu des polices dans Flash contrôle l'apparence de votre texte dans un fichier SWF ; c'est-à-dire la manière dont il est rendu (ou *tracé*) à l'exécution. La technologie de rendu avancé des polices utilisée dans Flash Player 8 et les versions ultérieures est appelée anti-alias avancé. L'anti-alias avancé utilise la technologie de rendu avancé pour contribuer à rendre le texte lisible et clair même avec les polices de taille réduite ou moyenne, par exemple lorsque vous appliquez l'anti-alias avancé à vos champs de texte. Cette technologie est présentée plus en détail ci-après dans cette section.

L'anti-alias vous permet de lisser le texte de sorte que les contours des caractères affichés à l'écran soient plus réguliers. Cette fonctionnalité est très utile pour afficher du texte de petite taille. L'option anti-alias pour le texte rend les caractères plus lisibles en alignant le contour du texte le long des limites de pixels ; elle est particulièrement efficace pour obtenir un rendu plus clair des polices de petite taille. Vous pouvez appliquer l'anti-alias avancé à chaque champ de texte de votre application, plutôt qu'à des caractères individuels.

L'anti-alias avancé est pris en charge pour le texte statique, dynamique et de saisie lorsque l'utilisateur est équipé de Flash Player 7 ou d'une version ultérieure. Avec une version antérieure de Flash Player, elle ne fonctionne que pour le texte statique. Les options d'anti-alias avancé sont disponibles pour Flash Player 8 et les versions ultérieures.

Flash intègre une technologie de rastérisation et de rendu des polices nettement améliorée appelée anti-alias avancé, qui permet de travailler avec des polices anti-aliasées. Flash comprend cinq méthodes de rendu des polices, lesquelles ne sont disponibles que lorsque vous publiez des fichiers SWF pour Flash Player 8 et les versions ultérieures. Si vous publiez des fichiers à utiliser avec Flash Player 7 ou des versions antérieures, seule l'option Anti-Alias pour l'animation est disponible pour vos champs de texte.

L'anti-alias avancé est une technologie de rendu des polices de grande qualité que vous pouvez activer en utilisant l'outil de programmation de Flash ou du code ActionScript. La technologie anti-alias avancé vous permet de bénéficier d'un rendu des polices de grande qualité même pour les polices de petite taille, en disposant d'un meilleur contrôle. Vous pouvez appliquer l'anti-alias avancé au rendu des polices intégrées pour les champs de texte statiques, dynamiques et de saisie. Les capacités améliorées signifient que le texte intégré apparaît au même niveau de qualité que le texte de périphérique et que les polices présentent le même aspect sur différentes plates-formes.

Les méthodes de rendu des polices disponibles pour Flash Player 8 et les versions ultérieures sont Polices de périphérique, Texte bitmap (sans anti-alias), Anti-alias pour l'animation, Anti-alias pour la lisibilité et Anti-alias personnalisé, qui vous permet de définir une valeur personnalisée pour l'épaisseur et la netteté. Pour plus d'informations sur ces options, consultez « Options de rendu des polices dans Flash », à la page 397.

REMARQUE

Lorsque vous ouvrez des fichiers FLA existants dans Flash 8 et les versions ultérieures, votre texte n'est pas automatiquement mis à jour vers l'option Anti-alias pour la lisibilité ; vous devez sélectionner des champs de texte individuels et modifier manuellement les paramètres d'anti-alias afin de bénéficier de la technologie de rendu anti-alias avancé.

Les fonctionnalités d'anti-alias avancé et personnalisé prennent en charge les éléments suivants :

- Mise à l'échelle et rotation de texte
- Toutes les polices (normal, gras ou italiques) jusqu'à la taille de 255 points
- L'exportation de fichiers vers la plupart des formats (tels que les fichiers JPEG ou GIF)

Les fonctionnalités d'anti-alias avancé et personnalisé ne prennent pas en charge les éléments suivants :

- Flash Player 7 ou versions antérieures
- Texte incliné ou renversé
- Impression
- Exportation de fichiers vers le format PNG

REMARQUE

Lorsque le texte est animé, le lecteur désactive l'anti-alias avancé afin d'améliorer l'apparence de votre texte pendant son déplacement. Une fois l'animation terminée, l'anti-alias est réactivé.

Pour un exemple de fichier source, aliasing.fla, qui décrit comment appliquer et manipuler du texte anti-aliasé dans une application, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Advanced Anti-Aliasing afin d'accéder à cet exemple. Vous utilisez la technologie de rendu anti-alias avancé pour créer un petit texte néanmoins très lisible. Cet exemple présente également le défilement rapide et fluide des champs de texte à l'aide de la propriété `cacheAsBitmap`.

Options de rendu des polices dans Flash

Cins options différentes de rendu des polices sont disponibles dans Flash. Pour sélectionner une option, choisissez le champ de texte et ouvrez l'inspecteur des propriétés. Sélectionnez une option dans le menu contextuel de Méthode de rendu des polices.

Polices de périphérique Produit un fichier SWF de plus petite taille. Cette option assure le rendu en utilisant les polices actuellement installées sur l'ordinateur de l'utilisateur final.

Texte Bitmap (sans anti-alias) produit un texte aux bords nets, sans anti-alias. Cette option génère un fichier SWF plus volumineux, car le contour des polices est inclus dans le fichier SWF.

Anti-Alias pour l'animation Produit un texte anti-alias à l'animation fluide. L'animation du texte est également plus rapide dans certains cas, car l'alignement et l'anti-aliasing ne sont pas appliqués pendant l'animation. Vous ne constatez pas d'amélioration des performances lorsque vous utilisez des polices de grande taille comprenant beaucoup de lettres ou des polices mises à l'échelle. Cette option génère un fichier SWF plus volumineux, car le contour des polices est inclus dans le fichier SWF.

Anti-Alias pour la lisibilité Le moteur d'anti-alias avancé est utilisé pour cette option. Elle offre le plus haut niveau de qualité de texte et la meilleure lisibilité. Elle produit aussi le fichier SWF le plus gros car ce dernier inclut les contours des polices et des informations spéciales sur l'anti-alias.

Anti-alias personnalisé identique à l'anti-alias pour la lisibilité, si ce n'est que vous pouvez manipuler visuellement les paramères d'anti-alias avancé pour obtenir un aspect spécifique. Cette option est utile pour donner la meilleure apparence possible aux polices nouvelles ou rares.

Pour obtenir un exemple d'utilisation de l'anti-alias avec ActionScript, consultez « Définition de l'anti-alias avec ActionScript », à la page 398.

A propos de la modulation continue des traits

La technologie de rendu des polices anti-alias avancé exploite les propriétés inhérentes aux champs de distance afin de proposer une modulation continue des traits (CSM) ; par exemple, la modulation continue de l'épaisseur des traits et de la netteté des contours du texte. CSM utilise deux paramètres de rendu pour contrôler le mappage des distances des champs de distance échantillonnées de manière adaptative (ADF) par rapport aux vaneurs de densité des glyphes. Les valeurs optimales de ces paramètres sont extrêmement subjectives ; elles peuvent dépendre des préférences de l'utilisateur, des conditions d'éclairage, des propriétés de l'affichage, de la police, des couleurs de premier plan et d'arrière-plan et de la taille des points. La fonction qui mappe les distances ADF par rapport aux valeurs de densité présente une limite extérieure sous laquelle les valeurs sont définies sur 0 et une limite intérieure au-dessus de laquelle les valeurs sont réglées sur une valeur de densité maximale, telle que 255.

Définition de l'anti-alias avec ActionScript

Flash propose deux types d'anti-alias : normal et avancé. L'anti-alias avancé est disponible uniquement dans Flash Player 8 et les versions ultérieures et peut uniquement être utilisé si vous intégrez la police dans la bibliothèque et si la propriété `embedFonts` du champ de texte est définie sur `true`. Pour Flash Player 8 et les versions ultérieures, le paramètre par défaut des champs de texte créés à l'aide d'ActionScript est `normal`.

Pour définir les valeurs de la propriété `TextField.antiAliasType`, utilisez les valeurs de chaîne suivantes :

normal Applique l'anti-alias de texte normal. Cela correspond au type d'anti-alias utilisé par Flash Player dans la version 7 et les versions antérieures.

avancé Applique l'anti-alias avancé pour améliorer la lisibilité du texte, qui est disponible dans Flash Player 8 et les versions ultérieures. L'anti-alias avancé permet un rendu de grande qualité des polices de petite taille. Il est préférable de l'utiliser avec les applications comprenant une grande quantité de texte de petite taille.

CONSEIL	Adobe ne recommande pas l'anti-alias avancé pour les polices de plus de 48 points.

Pour utiliser ActionScript afin de définir le texte d'anti-alias, consultez l'exemple suivant.

Pour utiliser l'anti-alias avancé :

1. Créez un document Flash et enregistrez-le sous le nom de **antialiastype.fla**.
2. Créez deux clips sur la scène et attribuez-leur les noms d'occurrences **normal_mc** et **advanced_mc**.

   Vous utiliserez ces clips pour alterner entre les deux types d'anti-alias : normal et avancé.
3. Ouvrez le panneau Bibliothèque et sélectionnez Nouvelle police dans le menu contextuel dans le coin supérieur droit du panneau Bibliothèque.

   La boîte de dialogue Propriétés des symboles de police s'ouvre et vous permet de sélectionner une police à intégrer dans le fichier SWF (y compris le style et la taille de la police). Vous pouvez également attribuer un nom de police qui apparaît dans la bibliothèque du document et dans le menu déroulant Polices de l'inspecteur des propriétés (si un champ de texte est sélectionné sur la scène).
   a. Sélectionnez la police Arial dans le menu déroulant Polices.
   b. Assurez-vous que les options Gras et Italique ne sont pas sélectionnées.
   c. Définissez la taille sur 10 pixels.
   d. Saisissez le nom **Arial-10 (embedded)**.
   e. Cliquez sur OK.
4. Dans la bibliothèque, cliquez avec le bouton droit de la souris sur le symbole de la police et sélectionnez Liaison dans le menu contextuel.

   La boîte de dialogue Propriétés de liaison apparaît.
5. Sélectionnez les options Exporter pour ActionScript et Exporter dans la première image, saisissez l'identifiant de liaison **Arial-10** et cliquez sur OK.
6. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
var text_fmt:TextFormat = new TextFormat();
text_fmt.font = "Arial-10";
text_fmt.size = 10;

this.createTextField("my_txt", 10, 20, 20, 320, 240);
my_txt.autoSize = "left";
my_txt.embedFonts = true;
my_txt.selectable = false;
my_txt.setNewTextFormat(text_fmt);
my_txt.multiline = true;
my_txt.wordWrap = true;
```

```
var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function(src:String) {
  if (src != undefined) {
    my_txt.text = src;
  } else {
    my_txt.text = "unable to load text file.";
  }
};
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");

normal_mc.onRelease = function() {
  my_txt.antiAliasType = "normal";
};
advanced_mc.onRelease = function() {
  my_txt.antiAliasType = "advanced";
};
```

Le code précédent est divisé en quatre zones clés. Le premier bloc de code crée un nouvel objet TextFormat, qui spécifie une police et une taille de police à utiliser pour un champ de texte qui sera créé prochainement. La police spécifiée, Arial-10, est l'identifiant de liaison pour le symbole de police que vous avez intégré lors d'une étape précédente.

Le deuxième bloc de code crée un nouveau champ de texte portant le nom d'occurrence `my_txt`. Pour que la police soit correctement intégrée, vous devez définir `embedFonts` sur `true` pour l'occurrence du champ de texte. Le code définit également le formatage du texte pour le nouveau champ de texte en fonction de l'objet TextFormat créé antérieurement.

Le troisième bloc de code définit une occurrence de LoadVars qui remplit le champ de texte de la scène avec le contenu d'un fichier de texte externe. Une fois le document entièrement chargé (mais pas analysé), tout le contenu du fichier est copié dans la propriété `my_txt.text`, de sorte qu'il apparaît sur la scène.

Le quatrième et dernier bloc de code définit les gestionnaires d'événements `onRelease` pour le clip `normal_mc` et le clip `advanced_mc`. Lorsque l'utilisateur clique et libère l'une de ces options, le type d'anti-alias correspondant au champ de texte de la scène est modifié.

7. Enregistrez vos modifications dans le fichier FLA.
8. Choisissez Contrôle > Tester l'animation pour tester votre document Flash.

9. Cliquez sur le clip `advanced_mc` sur la scène.

   Un clic sur le clip fait basculer le type d'anti-alias de normal (la valeur par défaut) à avancé. Lorsque vous traitez des champs de texte contenant une taille de police plus petite, la définition de l'anti-alias sur avancé peut améliorer considérablement la lisibilité du texte.

CONSEIL	L'anti-alias avancé offre un rendu de haute qualité pour les polices de petite taille. Il est préférable de l'utiliser avec les applications comprenant une grande quantité de texte de petite taille. Adobe ne recommande pas l'anti-alias avancé pour les polices de plus de 48 points.

Pour plus d'informations sur la mise en forme de texte anti-alias, consultez « Utilisation d'un type d'insertion dans la grille », à la page 408 et « A propos de la mise en forme du texte anti-alias », à la page 405.

Pour un exemple de fichier source, aliasing.fla, qui décrit comment appliquer et manipuler du texte anti-aliasé dans une application, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Advanced Anti-Aliasing afin d'accéder à cet exemple. Vous utilisez la technologie de rendu anti-alias avancé pour créer un petit texte néanmoins très lisible. Cet exemple présente également le défilement rapide et fluide des champs de texte à l'aide de la propriété `cacheAsBitmap`.

Définition des tableaux pour les polices

Si vous créez des polices à utiliser dans les fichiers SWF ou à distribuer aux développeurs Flash, vous devrez peut-être définir des tableaux de polices afin de contrôler leur rendu sur la scène.

L'anti-alias avancé utilise des champs de distance échantillonnés de manière adaptative (ADF) pour représenter lescontours déterminant un glyphe (un caractère). Flash utilise deux valeurs :

- Une valeur de limite extérieure, sous laquelle les densités sont définies sur 0.
- Une valeur de limite intérieure, au-dessus de laquelle les densités sont définies sur une densité maximale, telle que 255.

Entre ces deux valeurs limites, la fonction de mappage est une courbe linéaire qui varie entre 0 à la limite extérieure et la densité maximale à la limite intérieure.

L'ajustement des valeurs limites extérieure et intérieure affecte le poids des traits et la netteté des bords. L'espacement entre ces deux paramètres est comparable à deux fois le rayon de filtre des méthodes d'anti-alias classiques ; un espacement étroit fournit un bord plus net, alors qu'un espacement plus important offre des bords plus doux et plus filtrés. Lorsque l'espacement est de 0, l'image de densité produite est un bitmap à deux niveaux. Lorsque l'espacement est très large, l'image de densité produite présente des bords évoquant une aquarelle.

Généralement, les utilisateurs préfèrent les bords nets et très contrastés pour les polices de petite taille et des bords plus doux pour les textes animés et les polices plus grandes.

La limite extérieure possède généralement une valeur négative, la limite intérieure présentant une valeur positive et le point intermédiaire est proche de 0. L'ajustement de ces paramètres pour déplacer le point intermédiaire vers l'infini négatif augmente le poids du trait ; son déplacement vers l'infini négatif réduit le poids du trait.

REMARQUE	La limite extérieure devrait toujours être inférieure ou égale à la limite intérieure.

Flash Player comprend des paramètres d'anti-alias avancés pour dix polices de base ; pour ces dernières, les paramètres d'anti-alias avancé sont uniquement fournis pour les tailles de police de 6 à 20. Pour ces polices, toutes les tailles inférieures à 6 utilisent les paramètres liés à la taille 6, alors que les tailles supérieures à 20 utilisent les paramètres liés à la taille 20. Les autres polices sont mappées vers les données des polices fournies. La méthode `setAdvancedAntialiasingTable()` vous permet de définir des données d'anti-alias personnaliisées pour d'autres polices et d'autres tailles de polices, ou de remplacer les paramètres par défaut des polices fournies. Pour plus d'informations sur la création d'un tableau d'anti-alias, reportez-vous à l'exemple suivant :

Pour créer un tableau d'anti-alias avancé pour une police intégrée :

1. Créez un document Flash, puis enregistrez-le sous le nom de **advancedaatable.fla**.
2. Sélectionnez Nouvelle police dans le menu contextuel Bibliothèque.
3. Sélectionnez Arial dans le menu contextuel Police, puis définissez la taille de la police sur **32** points.
4. Sélectionnez les options Gras et Italique.
5. Saisissez le nom de police **Arial (embedded)** dans le champ Nom et cliquez sur OK.

6. Cliquez avec le bouton droit de la souris (Windows) ou appuyez sur la touche Contrôle (Macintosh) dans la Bibliothèque, et choisissez Liaison.
7. Dans la boîte de dialogue Propriétés de liaison :
   a. Tapez **Arial-embedded** dans le champ Identifiant.
   b. Sélectionnez Exporter pour ActionScript et Exporter dans la première image.
   c. Cliquez sur OK.
8. Sélectionnez Image 1 pour le scénario principal, puis ajoutez l'ActionScript suivant dans le panneau Actions :

```
import flash.text.TextRenderer;
var arialTable:Array = new Array();
arialTable.push({fontSize:16.0, insideCutoff:0.516,
  outsideCutoff:0.416});
arialTable.push({fontSize:32.0, insideCutoff:2.8, outsideCutoff:-2.8});
TextRenderer.setAdvancedAntialiasingTable("Arial", "bolditalic", "dark",
  arialTable);

var my_fmt:TextFormat = new TextFormat();
my_fmt.align = "justify";
my_fmt.font = "Arial-embedded";
my_fmt.size = 32;

this.createTextField("my_txt", 999, 10, 10, Stage.width-20,
  Stage.height-20);
my_txt.antiAliasType = "advanced";
my_txt.embedFonts = true;
my_txt.multiline = true;
my_txt.setNewTextFormat(my_fmt);
my_txt.sharpness = 0;
my_txt.thickness = 0;
my_txt.wordWrap = true;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function(src:String):Void {
  if (src != undefined) {
    my_txt.text = src + "\n\n" + src;
  } else {
    trace("error downloading text file");
  }
};
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

Le code précédent est divisé en quatre sections. La première section de code importe la classe TextRenderer et définit un nouveau tableau d'anti-alias pour deux tailles différentes de la police Arial. La deuxième section de code définit un nouvel objet TextFormat, que vous utiliserez pour appliquer le formatage de texte au champ de texte (que vous créez dans la section de code suivante). La section de code suivante crée un nouveau champ de texte portant le nom d'occurrence `my_txt`, active l'anti-alias avancé, applique l'objet de format de texte (créé antérieurement) et active le texte multiligne et le retour à la ligne. Le dernier bloc de code définit un objet LoadVars que vous utilisez pour charger du texte à partir d'un fichier de texte externe et remplir le champ de texte sur la scène.

9. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

   Une fois le texte chargé à partir du serveur distant, Flash affiche du texte dans le champ de texte, ce qui vous permet d'observer l'effet des propriétés du tableau d'anti-alias avancé sur votre champ de texte. La police intégrée sur la scène devrait apparaître légèrement floue en raison des valeurs actuelles de `insideCutoff` et `outsideCutoff`.

A propos de la présentation et de la mise en forme du texte

Vous pouvez contrôler la présentation et la mise en forme du texte à l'aide d'ActionScript. La classe TextFormat permet de bien contrôler l'apparence du texte à l'exécution, en plus des autres formes de mise en forme telles que les feuilles de style (consultez « Formatage de texte avec les feuilles de style CSS », à la page 413) et le texte HTML (consultez « Utilisation de texte au format HTML », à la page 427).

Vous pouvez également contrôler l'insertion des caractères dans la grille à l'aide d'ActionScript lorsque vous utilisez le texte anti-alias dans un fichier SWF. Cela vous permet de contrôler l'apparence des caractères à l'exécution. Pour obtenir un exemple de l'utilisation d'un type d'insertion dans la grille de vos applications, consultez « Utilisation d'un type d'insertion dans la grille », à la page 408.

Pour plus d'informations sur les champs de texte, consultez « A propos des champs de texte », à la page 369. Pour plus d'informations sur la mise en forme du texte, consultez « A propos de la mise en forme du texte anti-alias », à la page 405. Pour plus d'informations sur la classe TextFormat, consultez « Utilisation de la classe TextFormat », à la page 410 et `TextFormat` dans le *Guide de référence du langage ActionScript 2.0.*

Pour plus d'informations sur la présentation et la mise en forme du texte à l'aide de la classe TextFormat, consultez les sections suivantes :

- « A propos de la mise en forme du texte anti-alias », à la page 405
- « Utilisation d'un type d'insertion dans la grille », à la page 408
- « Utilisation de la classe TextFormat », à la page 410
- « Propriétés par défaut des nouveaux champs de texte », à la page 412

A propos de la mise en forme du texte anti-alias

Flash présente deux propriétés que vous pourrez utiliser pour mettre en forme des champs de texte alors que l'anti-alias avancé est activé : `netteté` et `épaisseur`. la netteté fait référence à la quantité d'alias appliquée à l'occurrence du champ de texte. Une valeur de netteté élevée donne aux bords de la police intégrée un aspect découpé et net. La définition de la netteté sur une valeur inférieure donne à la police un aspect plus doux, un peu plus flou. La définition de l'épaisseur d'une police est analogue à l'activation de la mise en forme de gras pour un champ de texte. Plus l'épaisseur est importante, plus la police prend un aspect gras.

L'exemple suivant charge dynamiquement un fichier de texte et affiche le texte sur la scène. Le déplacement du pointeur de la souris le long de l'axe des *x* définit la netteté entre - 400 et 400. Le déplacement du pointeur de la souris le long de l'axe des *y* définit l'épaisseur entre - 200 et 200.

Pour modifier la netteté et l'épaisseur d'un champ de texte :

1. Créez un document Flash, puis enregistrez-le sous le nom de **sharpness.fla**.
2. Sélectionnez Nouvelle police dans le menu contextuel dans le coin supérieur droit du panneau Bibliothèque.
3. Sélectionnez Arial dans le menu déroulant Police, puis définissez la taille de la police sur 24 points.
4. Saisissez le nom de police **Arial-24 (embedded)** dans le champ Nom et cliquez sur OK.
5. Cliquez avec le bouton droit de la souris sur le symbole de la police dans la bibliothèque et sélectionnez Liaison pour ouvrir la boîte de dialogue Propriétés de liaison.
6. Définissez l'identifiant de liaison sur Arial-4, cochez les cases Exporter pour ActionScript et Exporter dans la première image, puis cliquez sur OK.

7. Ajoutez le code suivant à l'image 1 du scénario principal :

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.size = 24;
my_fmt.font = "Arial-24";

this.createTextField("lorem_txt", 10, 0, 20, Stage.width, (Stage.height
  - 20));
lorem_txt.setNewTextFormat(my_fmt);
lorem_txt.text = "loading...";
lorem_txt.wordWrap = true;
lorem_txt.autoSize = "left";
lorem_txt.embedFonts = true;
lorem_txt.antiAliasType = "advanced";

this.createTextField("debug_txt", 100, 0, 0, Stage.width, 20);
debug_txt.autoSize = "left";
debug_txt.background = 0xFFFFFF;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function(src:String) {
  lorem_txt.text = src;
}
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  lorem_txt.sharpness = (_xmouse * (800 / Stage.width)) - 400;
  lorem_txt.thickness = (_ymouse * (400 / Stage.height)) - 200;
  debug_txt.text = "sharpness=" + Math.round(lorem_txt.sharpness) +
    ", thickness=" + Math.round(lorem_txt.thickness);
};
Mouse.addListener(mouseListener);
```

Ce code ActionScript peut être divisé en cinq sections clés. La première section de code définit une nouvelle occurrence de TextFormat qui sera appliquée à un champ de texte créé dynamiquement. Les deux sections suivantes créent deux nouveaux champs de texte sur la scène. Le premier champ de texte, `lorem_txt`, applique l'objet de mise en forme de texte personnalisé créé antérieurement, active les polices intégrées et définit la propriété `antiAliasType` sur `true`. Le deuxième champ de texte, `debug_txt`, affiche les valeurs actuelles de netteté et d'épaisseur de la police pour le champ de texte `lorem_txt`.
La quatrième section de code crée un objet LoadVars, responsable du chargement du fichier de texte externe et de remplir le champ de texte `lorem_txt`. La cinquième et dernière section de code définit un écouteur de souris qui est appelé chaque fois que le pointeur de la souris se déplace sur la scène. Les valeurs actuelles de `sharpness` et d'`thickness` sont calculées en fonction de la position actuelle du pointeur de la souris sur la scène.
Les propriétés `sharpness` et `thickness` sont définies pour le champ de texte `lorem_txt` et les valeurs actuelles s'affichent dans le champ de texte `debug_txt`.

8. Sélectionnez Contrôle > Tester l'animation pour tester le document.

   Déplacez le pointeur de la souris le long de l'axe des *x* afin de modifier la netteté du champ de texte. Déplacez le pointeur de la souris de gauche à droite pour augmenter la netteté et donner au texte un aspect plus découpé. Déplacez le pointeur de la souris le long de l'axe des *y* afin de modifier l'épaisseur du champ de texte.

Pour plus d'informations sur l'utilisation de texte anti-alias dans un fichier SWF, consultez les rubriques « Définition de l'anti-alias avec ActionScript », à la page 398, « Options de rendu des polices dans Flash », à la page 397 et « Utilisation d'un type d'insertion dans la grille », à la page 408.

Pour un exemple de fichier source, aliasing.fla, qui décrit comment appliquer et manipuler du texte anti-aliasé dans une application, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Advanced Anti-Aliasing afin d'accéder à cet exemple. Vous utilisez la technologie de rendu anti-alias avancé pour créer un petit texte néanmoins très lisible. Cet exemple présente également le défilement rapide et fluide des champs de texte à l'aide de la propriété `cacheAsBitmap`.

Utilisation d'un type d'insertion dans la grille

Lorsque vous utilisez l'anti-alias avancé sur un champ de texte, trois types d'insertion dans une grille sont disponibles :

aucune L'insertion dans une grille n'est pas utilisée. Les lignes horizontales et verticales des glyphes ne sont pas forcées dans la grille des pixels. Ce paramètre est généralement utile pour l'animation et pour les polices de grande taille.

pixel Spécifie que les lignes horizontales et verticales fortes sont insérées dans la grille des pixels. Ce paramètre fonctionne uniquement avec les champs de texte alignés à gauche. Il offre généralement la meilleure lisibilité pour le texte aligné à gauche.

sous-pixel Spécifie que les lignes horizontales et verticales fortes sont insérées dans la grille des sous-pixels sur un écran à cristaux liquides. Le paramètre sous-pixel est généralement utile pour le texte dynamique aligné à droite ou centré, et permet souvent de trouver un bonne équilibre entre animation et qualité du texte.

L'exemple suivant présente la procédure à suivre pour définir un type d'insertion dans une grille dans un champ de texte à l'aide d'ActionScript.

Pour définir un type d'insertion dans une grille dans un champ de texte :

1. Créez un document Flash, puis enregistrez-le sous le nom de **gridfittype.fla**.
2. Sélectionnez Nouvelle police dans le menu contextuel dans le coin supérieur droit du panneau Bibliothèque.
3. Sélectionnez Arial dans le menu déroulant Police, puis définissez la taille de la police sur 10 points.
4. Saisissez le nom de police **Arial-10 (embedded)** dans le champ Nom et cliquez sur OK.
5. Cliquez avec le bouton droit de la souris sur le symbole de la police dans la bibliothèque et sélectionnez Liaison pour ouvrir la boîte de dialogue Propriétés de liaison.
6. Définissez l'identifiant de liaison sur Arial-10, puis cochez les cases Exporter pour ActionScript et Exporter dans la première image.
7. Cliquez sur OK.

8. Ajoutez le code suivant à l'image 1 du scénario principal :

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.size = 10;
my_fmt.font = "Arial-10";
var h:Number = Math.floor(Stage.height / 3);

this.createTextField("none_txt", 10, 0, 0, Stage.width, h);
none_txt.antiAliasType = "advanced";
none_txt.embedFonts = true;
none_txt.gridFitType = "none";
none_txt.multiline = true;
none_txt.setNewTextFormat(my_fmt);
none_txt.text = "loading...";
none_txt.wordWrap = true;

this.createTextField("pixel_txt", 20, 0, h, Stage.width, h);
pixel_txt.antiAliasType = "advanced";
pixel_txt.embedFonts = true;
pixel_txt.gridFitType = "pixel";
pixel_txt.multiline = true;
pixel_txt.selectable = false;
pixel_txt.setNewTextFormat(my_fmt);
pixel_txt.text = "loading...";
pixel_txt.wordWrap = true;

this.createTextField("subpixel_txt", 30, 0, h*2, Stage.width, h);
subpixel_txt.antiAliasType = "advanced";
subpixel_txt.embedFonts = true;
subpixel_txt.gridFitType = "subpixel";
subpixel_txt.multiline = true;
subpixel_txt.setNewTextFormat(my_fmt);
subpixel_txt.text = "loading...";
subpixel_txt.wordWrap = true;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function(src:String):Void {
  if (src != undefined) {
    none_txt.text = "[antiAliasType=none]\n" + src;
    pixel_txt.text = "[antiAliasType=pixel]\n" + src;
    subpixel_txt.text = "[antiAliasType=subpixel]\n" + src;
  } else {
    trace("unable to load text file");
  }
};
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

Le code ActionScript précédent peut être divisé en cinq sections. La première section définit un nouvel objet de format de texte spécifiant deux propriétés, `size` et `font`. La propriété `font` fait référence à l'identifiant de liaison du symbole de police figurant actuellement dans la bibliothèque du document. Les deuxième, troisième et quatrième sections de code créent chacune un nouveau champ de texte dynamique sur la scène et définissent certaines propriétés communes : `antiAliasType` (qui doit être défini sur `avancé`), `embedFonts` (défini sur `true`), `multiline` et `wordWrap`. Chaque section applique également l'objet de format de texte créé dans une section précédente et définit le type d'insertion dans une grille sur `normal`, `pixel` ou `sous-pixel`. La cinquième et dernière section crée une occurrence de LoadVars, qui charge le contenu d'un fichier de texte externe dans chacun des champs de texte que vous créez avec du code.

9. Enregistrez le document et choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

   Chaque champ de texte doit être initialisé avec la valeur « `chargement...` ». Une fois le fichier de texte externe chargé avec succès, chaque champ de texte affiche un exemple de texte formaté utilisant un type différent d'insertion dans une grille.

CONSEIL	La technologie anti-alias avancé utilise l'insertion dans une grille uniquement si la rotation est de 0°.

Utilisation de la classe TextFormat

Vous pouvez utiliser la classe TextFormat pour définir les propriétés de formatage d'un champ de texte. Cette classe intègre des informations sur le formatage des caractères et des paragraphes. Les informations sur le formatage des caractères décrivent l'apparence des différents caractères : nom de police, taille, couleur et URL associée. Les informations sur le formatage des paragraphes décrivent l'apparence d'un paragraphe : marge de gauche, marge de droite, indentation de la première ligne, ainsi qu'alignement à gauche, à droite ou au centre.

Pour utiliser la classe TextFormat, vous devez d'abord créer un objet TextFormat et définir ses styles de formatage de caractères et de paragraphes. Appliquez ensuite l'objet TextFormat à un champ de texte à l'aide des méthodes `TextField.setTextFormat()` ou `TextField.setNewTextFormat()`.

La méthode `setTextFormat()` modifie le format de texte appliqué à chaque caractère, à des groupes de caractères ou à tout le texte d'un champ de texte. Cependant, le texte nouvellement inséré (saisi par l'utilisateur ou inséré par du code ActionScript) n'adopte pas le formatage spécifié par un appel de la méthode `setTextFormat()`. Pour spécifier le formatage par défaut d'un texte nouvellement inséré, utilisez `TextField.setNewTextFormat()`. Pour plus d'informations, consultez les sections setTextFormat (méthode TextField.setTextFormat) et setNewTextFormat (méthode TextField.setNewTextFormat) dans le *Guide de référence du langage ActionScript 2.0.*

Pour formater un champ de texte avec la classe TextFormat :

1. Dans un nouveau document Flash, créez un champ de texte sur la scène à l'aide de l'outil Texte.

   Tapez du texte dans le champ de texte sur la scène ; par exemple **Texte gras, italique, 24 points**.

2. Dans l'inspecteur des propriétés, tapez **myText_txt** dans la zone de texte Nom de l'occurrence, sélectionnez Dynamique dans le menu contextuel Type de texte, puis sélectionnez Multiligne dans le menu contextuel Type de ligne.

3. Sélectionnez l'image 1 dans le scénario et ouvrez le panneau Actions (Fenêtre>Actions).

4. saisissez le code suivant dans le panneau Actions pour créer un objet TextFormat et définissez ses propriétés `bold` et `italic` sur la valeur `true` et sa propriété `size` sur 24.

   ```
   // Créez un objet TextFormat.
   var txt_fmt:TextFormat = new TextFormat();
   // Spécifiez le formatage des paragraphes et des caractères.
   txt_fmt.bold = true;
   txt_fmt.italic = true;
   txt_fmt.size = 24;
   ```

5. Appliquez l'objet TextFormat au champ de texte que vous avez créé à l'étape 1 en utilisant `TextField.setTextFormat()`.

   ```
   myText_txt.setTextFormat(txt_fmt);
   ```

   Cette version de `setTextFormat()` applique le formatage spécifié à l'intégralité du champ de texte. Deux autres versions de cette méthode vous permettent d'appliquer le formatage à des caractères individuels ou à des groupes de caractères. Par exemple, le code suivant applique le formatage gras, italique, 24 points aux trois premiers caractères que vous avez entrés dans le champ texte :

   ```
   myText_txt.setTextFormat(0, 3, txt_fmt);
   ```

   Pour plus d'informations, consultez les sections setTextFormat (méthode TextField.setTextFormat) et setNewTextFormat (méthode TextField.setNewTextFormat) dans le *Guide de référence du langage ActionScript 2.0.*

6. Choisissez Contrôle > Tester l'animation pour tester l'application.

Pour plus d'informations sur l'utilisation de la classe TextFormat, consultez les rubriques suivantes :

- « Propriétés par défaut des nouveaux champs de texte », à la page 412
- « Formatage de texte avec les feuilles de style CSS », à la page 413

Propriétés par défaut des nouveaux champs de texte

Les champs de texte créés à l'exécution avec `createTextField()` reçoivent un objet TextFormat par défaut avec les propriétés suivantes :

```
align = "left"
blockIndent = 0
bold = false
bullet = false
color = 0x000000
font = "Times New Roman" (default font is Times on Mac OS X)
indent = 0
italic = false
kerning = false
leading = 0
leftMargin = 0
letterSpacing = 0
rightMargin = 0
size = 12
tabStops = [] (empty array)
target = ""
underline = false
url = ""
```

REMARQUE	La propriété de police par défaut sur Mac OS X est Times.

Pour obtenir une liste complète des méthodes TextFormat et leur description, consultez `TextFormat` dans le *Guide de référence du langage ActionScript 2.0*.

Formatage de texte avec les feuilles de style CSS

Les styles CSS (feuilles de style en cascade) permettent de travailler avec des styles de texte pouvant être appliqués à des documents HTML ou XML. Une feuille de style est un ensemble de règles de formatage qui spécifie comment formater des éléments HTML ou XML. Chaque règle associe un nom de style, ou *sélecteur*, à une ou plusieurs propriétés de style ainsi qu'à leurs valeurs. Par exemple, le style suivant définit un sélecteur appelé `bodyText`.

```
.bodyText {
  text-align: left
}
```

Vous pouvez créer des styles définissant les balises de formatage HTML intégrées que Flash Player utilise (telles que `<p>` et `<li>`). Vous pouvez également créer des *classes* de style qui peuvent être appliquées à des éléments HTML spécifiques à l'aide de l'attribut `<p>` ou `<span>` des `classes` de balises, ou définir de nouvelles balises.

Utilisez la classe TextField.StyleSheet pour utiliser les feuilles de style de texte. Bien que la classe TextField puisse être utilisée avec Flash Player 6, la classe TextField.StyleSheet nécessite des fichiers SWF conçus pour Flash Player 7 ou une version plus récente. Vous pouvez charger des styles à partir d'un fichier CSS externe ou les créer de façon native en utilisant ActionScript. Pour appliquer une feuille de style à un champ de texte contenant du texte au format HTML ou XML, utilisez la propriété `TextField.styleSheet`. Les styles définis dans la feuille de style sont automatiquement mis en correspondance avec les balises définies dans le document HTML ou XML.

Pour utiliser des feuilles de style, vous devez suivre trois étapes essentielles :

- Créez un objet de feuille de style à partir de la classe TextField.StyleSheet (pour plus d'informations, consultez la section StyleSheet (TextField.StyleSheet) dans le *Guide de référence du langage ActionScript 2.0*).
- Ajoutez des styles à l'objet feuille de style, soit en les chargeant à partir d'un fichier CSS externe, soit en créant de nouveaux styles avec ActionScript.
- Affecter la feuille de style à un objet TextField contenant du texte au format XML ou HTML.

Pour plus d'informations, se reporter aux sections suivantes :

- « Propriétés CSS prises en charge », à la page 414
- « Création d'un objet feuille de style », à la page 416
- « Chargement de fichiers CSS externes », à la page 416
- « Création de nouveaux styles avec ActionScript », à la page 418

Pour un exemple de fichier source, formattedText.fla, qui décrit comment appliquer le formatage CSS au texte que vous chargez dans un fichier SWF à l'exécution, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/LoadText afin d'accéder à l'exemple.

Propriétés CSS prises en charge

Flash Player prend en charge un sous-ensemble de propriétés dans la spécification CSS1 d'origine (www.w3.org/TR/REC-CSS1). Le tableau suivant présente les propriétés et les valeurs CSS prises en charge et les noms de propriétés ActionScript correspondants. (Chaque nom de propriété ActionScript est tiré du nom de propriété CSS correspondant. Le trait d'union est omis et le caractère suivant est une majuscule.)

Propriété CSS	Propriété ActionScript	Utilisation et valeurs prises en charge
text-align	textAlign	Les valeurs reconnues sont `left`, `center`, right et `justify`.
font-size	fontSize	Seule la partie numérique de la valeur est utilisée. Les unités (px, pt) ne sont pas analysées ; les pixels et les points sont équivalents.
text-decoration	textDecoration	Les valeurs reconnues sont `none` et `underline`.
margin-left	marginLeft	Seule la partie numérique de la valeur est utilisée. Les unités (px, pt) ne sont pas analysées ; les pixels et les points sont équivalents.
margin-right	marginRight	Seule la partie numérique de la valeur est utilisée. Les unités (px, pt) ne sont pas analysées ; les pixels et les points sont équivalents.
font-weight	fontWeight	Les valeurs reconnues sont `normal` et `bold`.

Propriété CSS	Propriété ActionScript	Utilisation et valeurs prises en charge
kerning	kerning	Les valeurs reconnues sont `true` et `false`.
font-style	fontStyle	Les valeurs reconnues sont `normal` et `italic`.
interlettrage	letterSpacing	Seule la partie numérique de la valeur est utilisée. Les unités (px, pt) ne sont pas analysées ; les pixels et les points sont équivalents.
text-indent	textIndent	Seule la partie numérique de la valeur est utilisée. Les unités (px, pt) ne sont pas analysées ; les pixels et les points sont équivalents.
font-family	fontFamily	Liste des polices à utiliser, séparées par des virgules, classées par ordre de choix décroissant. Tous les noms de familles de polices peuvent être utilisés. Si vous spécifiez un nom de police générique, il est converti dans la police de périphérique appropriée. Les conversions de police suivantes sont disponibles : `mono` est converti en `_typewriter`, `sans-serif` est converti en`_sans` et `serif` est converti en `_serif`.
color	color	Seules les valeurs hexadécimales de couleur sont prises en charge. Les couleurs désignées (comme `blue`) ne sont pas prises en charge. Les couleurs sont écrites au format suivant : `#FF0000`.
affichage	affichage	Les valeurs prises en charge sont `inline`, `block` et `none`.

Pour consulter un exemple d'utilisation de styles sur des éléments XML, reportez-vous à « Exemple d'utilisation de styles avec XML », à la page 425.

Création d'un objet feuille de style

Les styles CSS sont représentés dans ActionScript par la classe TextField.StyleSheet. Cette classe est uniquement disponible pour les fichiers SWF conçus pour Flash Player 7 et les versions ultérieures. Pour créer un objet feuille de style, appelez la fonction de constructeur de la classe TextField.StyleSheet.

```
var newStyle:TextField.StyleSheet = new TextField.StyleSheet();
```

Pour ajouter des styles à un objet feuille de style, chargez un fichier CSS externe dans l'objet ou définissez les styles dans ActionScript. Reportez-vous à « Chargement de fichiers CSS externes », à la page 416 et à « Création de nouveaux styles avec ActionScript », à la page 418.

Pour un exemple de fichier source, formattedText.fla, qui décrit comment appliquer le formatage CSS au texte que vous chargez dans un fichier SWF à l'exécution, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/LoadText afin d'accéder à l'exemple.

Chargement de fichiers CSS externes

Vous pouvez définir des styles dans un fichier CSS externe, puis charger ce fichier dans un objet feuille de style. Les styles définis dans le fichier CSS sont ajoutés à l'objet feuille de style. Pour charger un fichier CSS externe, utilisez la méthode `load()` de la classe TextField.StyleSheet. Pour déterminer le moment où le chargement du fichier CSS est terminé, utilisez le gestionnaire d'événement `onLoad` de l'objet feuille de style.

L'exemple suivant crée et charge un fichier CSS et utilise la méthode `TextField.StyleSheet.getStyleNames()` pour récupérer les noms des styles chargés.

Pour charger une feuille de style externe :

1. Créez un fichier dans l'éditeur de texte ou l'éditeur CSS de votre choix.
2. Ajoutez les définitions de style suivantes au fichier :

```
.bodyText {
  font-family: Arial,Helvetica,sans-serif;
  font-size: 12px;
}

.headline {
  font-family: Arial,Helvetica,sans-serif;
  font-size: 24px;
}
```

3. Enregistrez le fichier CSS sous styles.css.

4. Dans Flash, créez un fichier FLA.
5. Dans le scénario (Fenêtre > Scénario), sélectionnez le calque 1.
6. Ouvrez le panneau Actions (Fenêtre > Actions).
7. Ajoutez le code suivant au panneau Actions :

```
var styles:TextField.StyleSheet = new TextField.StyleSheet();
styles.onLoad = function(success:Boolean):Void {
  if (success) {
    // afficher les noms de style.
    trace(this.getStyleNames());
  } else {
    trace("Error loading CSS file.");
  }
};
styles.load("styles.css");
```

REMARQUE

Dans l'extrait de code précédent, `this.getStyleNames()` fait référence à l'objet `styles` que vous avez construit dans la première ligne d'ActionScript.

8. Enregistrez le fichier FLA dans le répertoire qui contient styles.css.
9. Testez le document Flash (Contrôle > Tester l'animation).

   Les noms des deux styles devraient s'afficher dans le panneau Sortie :

   ```
   .bodyText,.headline
   ```

   Si « Erreur lors du chargement du fichier CSS » s'affiche dans le panneau Sortie, assurez-vous que les fichiers FLA et CSS se trouvent dans le même répertoire et que vous avez tapé le nom du fichier CSS correctement.

Comme pour les autres méthodes ActionScript qui chargent des données via le réseau, le fichier CSS doit résider dans le même domaine que le fichier SWF qui effectue le chargement du fichier. (Voir la section « Restriction des API de réseau », à la page 695.) Pour plus d'informations sur l'utilisation des styles CSS avec Flash, consultez la section StyleSheet (TextField.StyleSheet) dans le *Guide de référence du langage ActionScript 2.0*.

Pour un exemple de fichier source, formattedText.fla, qui décrit comment appliquer le formatage CSS au texte que vous chargez dans un fichier SWF à l'exécution, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/LoadText afin d'accéder à l'exemple.

Création de nouveaux styles avec ActionScript

Vous pouvez créer de nouveaux styles de texte avec ActionScript en utilisant la méthode `setStyle()` de la classe TextField.StyleSheet. Cette méthode prend deux paramètres : le nom du style et un objet qui définit les propriétés de ce style.

Par exemple, le code suivant crée un objet feuille de style nommé `styles` qui définit deux styles identiques à ceux que vous avez déjà importés (consultez « Chargement de fichiers CSS externes », à la page 416).

```
var styles:TextField.StyleSheet = new TextField.StyleSheet();
styles.setStyle("bodyText",
  {fontFamily: 'Arial,Helvetica,sans-serif',
  fontSize: '12px'}
);
styles.setStyle("headline",
  {fontFamily: 'Arial,Helvetica,sans-serif',
  fontSize: '24px'}
);
```

Application de styles à un objet TextField

Pour appliquer un objet feuille de style à un objet TextField, affectez cet objet à la propriété `styleSheet` de l'objet champ de texte.

```
textObj_txt.styleSheet = styles;
```

REMARQUE

Ne confondez pas la *propriété* `TextField.styleSheet` avec la *classe* TextField.StyleSheet. Les majuscules les distinguent.

Lorsque vous affectez un objet feuille style à un objet TextField, le comportement habituel du champ de texte est modifié de la manière suivante :

- Les propriétés `text` et `htmlText` du champ de texte, et toute variable qui lui est associée, contiennent toujours la même valeur et se comportent de la même façon.
- Le champ de texte est en lecture seule et ne peut plus être modifié par l'utilisateur.
- Les méthodes `setTextFormat()` et `replaceSel()` de la classe TextField ne fonctionnent plus avec le champ de texte. La seule façon de modifier le champ consiste à changer la propriété `text` ou `htmlText` du champ de texte ou à modifier les variables associées au champ de texte.
- Tout texte affecté à la propriété `text` ou `htmlText` ou aux variables du champ de texte est stocké textuellement. Tout ce qui est écrit dans l'une de ces propriétés peut être récupéré dans la forme originale du texte.

Application d'une feuille de style à un composant TextArea

Pour appliquer une feuille de style à un composant TextArea, vous devez créer un objet feuille de style et lui attribuer des styles HTML par le biais de la classe TextField.StyleSheet. Vous pouvez alors affecter la feuille de style à la propriété `styleSheet` du composant TextArea.

Les exemples suivants créent un objet feuille de style, `styles` et l'affectent à l'occurrence du composant `myTextArea`.

Utilisation d'une feuille de style avec un composant TextArea

1. Créez un document Flash, puis enregistrez-le sous le nom de **textareastyle.fla**.
2. Faites glisser un composant TextArea du dossier User Interface du panneau Composants sur la scène et attribuez-lui le nom d'occurrence **myTextArea**.
3. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
// Création d'un objet feuille de style et définition de ses styles.
var styles:TextField.StyleSheet = new TextField.StyleSheet();
styles.setStyle("html", {fontFamily:'Arial,Helvetica,sans-serif',
            fontSize:'12px',
            color:'#0000FF'});
styles.setStyle("body", {color:'#00CCFF',
            textDecoration:'underline'});
styles.setStyle("h1",{fontFamily:'Arial,Helvetica,sans-serif',
            fontSize:'24px',
            color:'#006600'});

/* Affectation de l'objet feuille de style au composant myTextArea.
  Définition de la propriété html sur true, définition de la propriété
  styleSheet sur l'objet feuille de style. */
myTextArea.styleSheet = styles;
myTextArea.html = true;

var myVars:LoadVars = new LoadVars();
// Définition du gestionnaire onData et chargement du texte à afficher.
myVars.onData = function(myStr:String):Void {
  if (myStr != undefined) {
    myTextArea.text = myStr;
  } else {
    trace("Unable to load text file.");
  }
};
myVars.load("http://www.helpexamples.com/flash/myText.htm");
```

Le bloc de code précédent crée une occurrence de TextField.StyleSheet qui définit trois styles : pour les balises HTML `html`, `body` et `h1`. L'objet feuille de style est ensuite appliqué au composant TextArea et le formatage HTML est activé. L'ActionScript restant définit un objet LoadVars qui charge un fichier HTML externe et remplit la zone de texte avec le texte chargé.

4. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

Association de styles

Les styles CSS dans Flash Player sont additionnels, c'est-à-dire que lorsque les styles sont imbriqués, chaque niveau d'imbrication peut fournir des informations de style, qui sont ajoutées les unes aux autres pour donner la mise en forme finale.

L'exemple suivant présente des données XML affectées à un champ de texte :

```
<sectionHeading>This is a section</sectionHeading>
<mainBody>This is some main body text, with one
<emphasized>emphatic</emphasized> word.</mainBody>
```

Pour le mot *emphatic* du texte ci-dessus, le style `emphasized` est imbriqué dans le style `mainBody`. Le style `mainBody` fournit les règles de couleur, de taille de police et de décoration. Le style `emphasized` ajoute une règle d'épaisseur de police à ces règles. Le mot *emphatic* sera formaté en associant les règles spécifiées par `mainBody` et `emphasized`.

Utilisation des classes de style

Vous pouvez créer des « classes » de style (ne constituant pas de véritables classes ActionScript 2.0) que vous pouvez appliquer à une balise `<p>` ou `<span>` par l'intermédiaire de l'attribut `class` de l'une de ces balises. Lorsqu'il est appliqué à une balise `<p>`, le style affecte tout le paragraphe. A l'aide de la balise `<span>`, vous pouvez également appliquer un style à une plage de texte qui utilise une classe de style.

Par exemple, la feuille de style suivante définit deux classes de style : `mainBody` et `emphasis` :

```
.mainBody {
  font-family: Arial,Helvetica,sans-serif;
  font-size: 24px;
}
.emphasis {
  color: #666666;
  font-style: italic;
}
```

A l'intérieur du texte HTML que vous affectez à un champ de texte, vous pouvez appliquer ces styles aux balises `<p>` et `<span>` comme expliqué ci-dessous :

```
<p class='mainBody'>This is <span class='emphasis'>really exciting!</
  span></p>
```

Définition du style de balises HTML intégrées

Flash Player prend en charge un sous-ensemble de balises HTML. (Pour plus d'informations, voir « Utilisation de texte au format HTML », à la page 427.) Vous pouvez affecter un style CSS à chaque occurrence d'une balise HTML intégrée qui apparaît dans un champ de texte. Par exemple, le code suivant définit un style pour la balise HTML intégrée `<p>`. Toutes les occurrences de cette balise voient leur style défini de la manière spécifiée par la règle de style :

```
p {
  font-family: Arial,Helvetica,sans-serif;
  font-size: 12px;
  display: inline;
}
```

Le tableau suivant indique les balises HTML intégrées dont le style peut être défini, et la façon dont chaque style est appliqué :

Nom de style	Mode d'application du style
`p`	Affecte toutes les balises `<p>`.
`body`	Affecte toutes les balises `<body>`. S'il est spécifié, le style `p` est prioritaire par rapport au style `body` .
`li`	Affecte toutes les balises à puce `<li>`.
`a`	Affecte toutes les balises d'ancrage `<a>`.
`a:link`	Affecte toutes les balises d'ancrage `<a>`. Ce style est appliqué après tout style `a`.
`a:hover`	Appliqué à une balise d'ancrage `<a>` lorsque le pointeur de la souris se déplace sur le lien. Ce style est appliqué après tout style `a` et `a:link`. Une fois que le pointeur la souris s'éloigne du lien, le style `a:hover` est supprimé du lien.
`a:active`	S'applique à une balise d'ancrage `<a>` lorsque l'utilisateur clique avec la souris sur le lien. Ce style est appliqué après tout style `a` et `a:link`. Une fois que le bouton de la souris est relâché, le style `a:active` est supprimé du lien.

Exemple d'utilisation de styles avec HTML

Cette section présente un exemple de l'utilisation des styles avec des balises HTML. Vous créez une feuille de style qui définit le style de certaines balises intégrées ainsi que certaines classes de style. Vous pouvez alors appliquer cette feuille de style à un objet TextField qui contient du texte au format HTML.

Pour formater du code HTML avec une feuille de style :

1. Créez un fichier dans l'éditeur de texte ou l'éditeur CSS de votre choix.
2. Ajoutez la définition de feuille de style suivante dans le fichier :

```
p {
  color: #000000;
  font-family: Arial,Helvetica,sans-serif;
  font-size: 12px;
  display: inline;
}

a:link {
  color: #FF0000;
}

a:hover{
  text-decoration: underline;
}

.headline {
  color: #000000;
  font-family: Arial,Helvetica,sans-serif;
  font-size: 18px;
  font-weight: bold;
  display: block;
}

.byline {
  color: #666600;
  font-style: italic;
  font-weight: bold;
  display: inline;
}
```

   Cette feuille de style définit des styles pour deux balises HTML intégrées (`<p>` et `<a>`) qui seront appliqués à toutes les occurrences de ces balises. Elle définit également deux classes de style (`.headline` et `.byline`) qui seront appliquées à des paragraphes et à des plages de texte spécifiques.
3. Enregistrez le fichier sous **html_styles.css**.

4. Créez un fichier texte dans un éditeur de texte ou HTML, puis enregistrez-le sous le nom de **myText.htm**.

   Ajoutez le texte suivant au fichier :

```
<p class='headline'>Flash adds advanced anti-aliasing rendering
  technology!</p><p><span class='byline'>San Francisco, CA</span>--Adobe
  Inc. announced today a new version of Flash that features a brand new
  font rendering technology called Advanced Anti-Aliasing, most
  excellent at rendering small text with incredible clarity and
  consistency across platforms. For more information, visit the <a
  href='http://www.adobe.com'>Adobe Flash web site.</a></p>
```

REMARQUE	Si vous copiez et collez cette chaîne de texte, veillez à supprimer les éventuels sauts de ligne qui peuvent y avoir été ajoutés.

5. Créez un document Flash dans l'outil auteur de Flash.
6. Sélectionnez la première image dans le calque 1 du scénario (Fenêtre > Scénario).
7. Ouvrez le panneau Actions (Fenêtre > Actions) et ajoutez-lui le code suivant :

```
this.createTextField("news_txt", 99, 50, 50, 450, 300);
news_txt.border = true;
news_txt.html = true;
news_txt.multiline = true;
news_txt.wordWrap = true;

// Création d'une feuille de style et d'un objet LoadVars.
var myVars_lv:LoadVars = new LoadVars();
var styles:TextField.StyleSheet = new TextField.StyleSheet();

// Emplacement des fichiers CSS et texte à charger.
var txt_url:String = "myText.htm";
var css_url:String = "html_styles.css";

// Définition du gestionnaire onLoad et chargement du fichier CSS.
styles.onLoad = function(success:Boolean):Void {
    if (success) {
        /* Si la feuille de style a été chargée sans erreur,
            l'affecter à l'objet texte,
            et affecter le texte HTML au champ de texte. */
        news_txt.styleSheet = styles;
} else {
        trace("Unable to load CSS file.");
    }
};
styles.load(css_url);
```

```
// Définition du gestionnaire onData et chargement du texte à afficher.
myVars_lv.onData = function(src:String):Void {
    if (src != undefined) {
        news_txt.htmlText = src;
    } else {
        trace("Unable to load HTML file");
    }
};
myVars_lv.load(txt_url);
```

REMARQUE

Dans cet ActionScript, vous chargez le texte à partir d'un fichier externe. Pour plus d'informations sur le chargement de données externes, consultez le Chapitre 14, « Utilisation des images, du son et de la vidéo ».

8. Enregistrez le fichier sous **news_html.fla** dans le même répertoire que celui contenant le fichier CSS que vous avez créé à l'étape 3.
9. Sélectionnez Contrôle > Tester l'animation pour voir les styles appliqués au texte HTML automatiquement.

Utilisation de styles pour définir de nouvelles balises

Si vous définissez un style dans une feuille de style, ce style peut être utilisé comme balise, comme vous utiliseriez une balise HTML intégrée. Par exemple, si une feuille de style définit un style CSS nommé `sectionHeading`, vous pouvez utiliser `<sectionHeading>` comme élément dans tout champ de texte associé à la feuille de style. Cette fonction vous permet d'affecter directement du texte aléatoire au format XML dans un champ de texte, afin que le texte soit automatiquement formaté en utilisant les règles de la feuille de style.

Par exemple, la feuille de style suivante crée les nouveaux styles `sectionHeading`, `mainBody` et `emphasized` :

```
sectionHeading {
  font-family: Verdana, Arial, Helvetica, sans-serif;
  font-size: 18px;
  display: block
}
mainBody {
  color: #000099;
  text-decoration: underline;
  font-size: 12px;
  display: block
}
emphasized {
  font-weight: bold;
  display: inline
}
```

Vous pouvez ensuite remplir un champ de texte associé à cette feuille de style avec le texte au format XML suivant :

```
<sectionHeading>This is a section</sectionHeading>
<mainBody>This is some main body text,
with one <emphasized>emphatic</emphasized> word.
</mainBody>
```

Exemple d'utilisation de styles avec XML

Dans cette section, vous créez un fichier FLA contenant du texte au format XML.
Vous créerez une feuille de style à l'aide du code ActionScript au lieu d'importer les styles à partir d'un fichier CSS comme illustré à la section « Exemple d'utilisation de styles avec HTML », à la page 422

Pour formater du XML avec une feuille de style :

1. Dans Flash, créez un fichier FLA.
2. A l'aide de l'outil Texte, créez un champ de texte d'environ 400 pixels de large et 300 pixels de haut.
3. Ouvrez l'inspecteur des propriétés (Fenêtre > Propriétés > Propriétés) et sélectionnez le champ de texte.
4. Dans l'inspecteur des propriétés, sélectionnez Texte dynamique dans le menu Type de texte, sélectionnez Multiligne dans le menu Type de ligne, sélectionnez l'option Rendre le texte au format HTML et tapez **news_txt** dans la zone de texte Nom de l'occurrence.
5. Dans le calque 1 du scénario (Fenêtre > Scénario), sélectionnez la première image.
6. Pour créer l'objet feuille de style, ouvrez le panneau Actions (Fenêtre > Actions) et ajoutez le code suivant au panneau Actions :

```
var styles:TextField.StyleSheet = new TextField.StyleSheet();
styles.setStyle("mainBody", {
  color:'#000000',
  fontFamily:'Arial,Helvetica,sans-serif',
  fontSize:'12',
  display:'block'
});
styles.setStyle("title", {
  color:'#000000',
  fontFamily:'Arial,Helvetica,sans-serif',
  fontSize:'18',
  display:'block',
  fontWeight:'bold'
});
```

```
styles.setStyle("byline", {
  color:'#666600',
  fontWeight:'bold',
  fontStyle:'italic',
  display:'inline'
});
styles.setStyle("a:link", {
  color:'#FF0000'
});
styles.setStyle("a:hover", {
  textDecoration:'underline'
});
```

Ce code crée un objet feuille de style nommé `xml_styles` qui définit les styles à l'aide de la méthode `setStyle()`. Les styles correspondent exactement à ceux que vous avez créés dans un fichier CSS externe, précédemment dans ce même chapitre.

7. Pour créer un texte XML à affecter au champ de texte, ouvrez un éditeur de texte et saisissez le texte suivant dans un nouveau document :

```
<story><title>Flash now has advanced anti-aliasing</
  title><mainBody><byline>San Francisco, CA</byline>--Adobe Inc.
  announced today a new version of Flash that features the new advanced
  anti-aliasing rendering technology. For more information, visit the <a
  href="http://www.adobe.com">Adobe Flash website</a></mainBody></
  story>
```

REMARQUE

Si vous copiez et collez cette chaîne de texte, veillez à supprimer les éventuels sauts de ligne qui peuvent y avoir été ajoutés. Sélectionnez Caractères masqués dans le menu contextuel du panneau Actions afin d'afficher et de supprimer les éventuels sauts de ligne supplémentaires.

8. Enregistrez le fichier texte sous **story.xml**.
9. Dans Flash, ajoutez le code suivant dans le panneau Actions, à la suite du code présenté à l'étape 6.

   Ce code charge le document story.xml, applique l'objet feuille de style à la propriété `styleSheet` du champ de texte et affecte le texte XML au champ de texte :

```
var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean):Void {
  if (success) {
    news_txt.styleSheet = styles;
    news_txt.text = my_xml;
  } else {
    trace("Error loading XML.");
  }
};
my_xml.load("story.xml");
```

REMARQUE	Vous chargez des données XML à partir d'un fichier externe dans cet ActionScript. Pour plus d'informations sur le chargement de données externes, consultez le Chapitre 14, « Utilisation des images, du son et de la vidéo ».

10. Enregistrez le fichier sous **news_xml.fla** dans le même dossier que le fichier story.xml.
11. Exécutez le fichier SWF (Contrôle > Tester l'animation) pour voir les styles automatiquement appliqués au texte dans le champ de texte.

Utilisation de texte au format HTML

Flash Player prend en charge un sous-ensemble de balises HTML comme `<p>` et `<li>` que vous pouvez utiliser pour attribuer un style à un texte dans un champ de texte dynamique ou de saisie. Dans Flash Player 7 et les versions ultérieures, les champs de texte prennent également en charge la balise `<img>`, qui vous permet d'intégrer des fichiers image (JPEG, GIF, PNG), des fichiers SWF et des clips dans un champ de texte. Dans Flash Player, le texte enveloppe automatiquement les images intégrées dans les champs de texte, de la même manière qu'un navigateur Web enveloppe le texte autour des images intégrées dans un document HTML. Pour plus d'informations, voir « Présentation de l'intégration d'images, de fichiers SWF et de clips dans les champs de texte », à la page 437.

Flash Player prend également en charge la balise `<textformat>`, qui vous permet d'appliquer les styles de formatage de paragraphe de la classe TextFormat aux champs de texte HTML. Pour plus d'informations, voir « Utilisation de la classe TextFormat », à la page 410.

Pour plus d'informations sur le formatage HTML du texte, consultez les rubriques suivantes :

- « Propriétés et syntaxe requises pour l'utilisation de texte au format HTML », à la page 428
- « Présentation des balises HTML prises en charge », à la page 429
- « Présentation des entités HTML prises en charge », à la page 436
- « Présentation de l'intégration d'images, de fichiers SWF et de clips dans les champs de texte », à la page 437

Propriétés et syntaxe requises pour l'utilisation de texte au format HTML

Pour utiliser du code HTML dans un champ texte, vous devez définir plusieurs propriétés, que ce soit avec l'inspecteur de propriétés ou avec le code ActionScript :

- Activez le formatage HTML du champ de texte en sélectionnant l'option Rendre le texte au format HTML dans l'inspecteur des propriétés ou en définissant la propriété `html` du champ de texte sur `true`.
- Pour utiliser des balises HTML, telles que `<p>`, `<br>` et `<img>`, vous devez transformer le champ de texte en champ multiligne en sélectionnant l'option Multiligne dans l'inspecteur des propriétés ou en définissant la propriété `multiline` du champ de texte sur `true`.
- Dans ActionScript, définissez la valeur de `TextField.htmlText` sur la chaîne de texte au format HTML que vous souhaitez afficher.

Par exemple, le code suivant active le formatage HTML pour un champ de texte nommé `titre_txt`, puis lui affecte du texte HTML.

```
this.createTextField("headline_txt", 1, 10, 10, 500, 300);
headline_txt.html = true;
headline_txt.wordWrap = true;
headline_txt.multiline = true;
headline_txt.htmlText = "<font face='Times New Roman' size='25'>This is how
  you assign HTML text to a text field.</font><br>It's very useful.</br>";
```

Pour rendre correctement le texte HTML, vous devez utiliser la syntaxe correcte. Les attributs des balises HTML doivent être encadrés de guillemets simples (') ou doubles ("). Les valeurs des attributs dépourvues de guillemets peuvent engendrer des résultats inattendus, comme le rendu incorrect du texte. Par exemple, le fragment de code HTML suivant *ne peut pas* être rendu correctement par Flash Player, car la valeur affectée à l'attribut `align` (`left`) n'est pas encadrée de guillemets :

```
this.createTextField("myField_txt", 10, 10, 10, 400, 200);
myField_txt.html = true;
myField_txt.htmlText = "<p align=left>This is left-aligned text</p>";
```

Si vous encadrez les valeurs d'attribut à l'aide de doubles guillemets, vous devez utiliser la fonction *d'échappement* pour les guillemets (`\"`). Les deux méthodes suivantes sont acceptables pour ce faire :

```
myField_txt.htmlText = "<p align='left'>This uses single quotes</p>";
myField_txt.htmlText = "<p align=\"left\">This uses escaped double quotes</
  p>";
myField_txt.htmlText = '<p align="left">This uses outer single quotes</p>';
myField_txt.htmlText = '<p align=\'left\'>This uses escaped single quotes</
  p>';
```

Il n'est pas nécessaire d'utiliser la fonction d'échappement pour les doubles guillemets si vous chargez du texte à partir d'un fichier externe. Cette action est uniquement nécessaire pour l'affectation des chaînes de texte dans ActionScript.

Présentation des balises HTML prises en charge

Cette section répertorie les balises HTML intégrées prises en charge par Flash Player. Vous pouvez également créer des styles et balises en utilisant les feuilles de style en cascade (CSS) (consultez « Formatage de texte avec les feuilles de style CSS », à la page 413).

Pour plus d'informations sur les balises HTML prises en charge, consultez les rubriques suivantes :

Balise d'ancrage

La balise `<a>` crée un hyperlien et prend en charge les attributs suivants :

- `href` Une chaîne comprenant 128 caractères au maximum et spécifiant l'URL de la page à charger dans le navigateur. L'URL peut être absolue ou relative à l'emplacement du fichier SWF qui charge la page. Par exemple, l'URL `http://www.adobe.com` constitue une référence absolue, alors que `/index.html` est une référence relative.
- `target` Spécifie le nom de la fenêtre cible dans laquelle vous chargez la page. Les options comprennent `_self`, `_blank`, `_parent` et `_top`. L'option `_self` spécifie l'image actuelle dans la fenêtre en cours, `_blank` spécifie une nouvelle fenêtre, `_parent` spécifie le parent de l'image actuelle et `_top` spécifie l'image de plus haut niveau dans la fenêtre en cours.

Par exemple, le fragment de code HTML suivant crée le lien Go home, qui ouvre www.adobe.com dans une nouvelle fenêtre du navigateur :

```
urlText_txt.htmlText = "<a href='http://www.adobe.com' target='_blank'>Go
  home</a>";
```

Vous pouvez utiliser le protocole spécial `asfunction` pour que le lien exécute une fonction ActionScript dans un fichier SWF au lieu d'ouvrir une URL. Pour plus d'informations sur le protocole `asfunction`, consultez la section relative au protocole asfunction dans le *Guide de référence du langage ActionScript 2.0*.

Vous pouvez également définir des styles `a:link`, `a:hover` et `a:active` pour les balises d'ancrage en utilisant les feuilles de style. Voir « Définition du style de balises HTML intégrées », à la page 421.

REMARQUE	Les URL absolues doivent avoir un préfixe `http://` ; dans le cas contraire, Flash les considère comme des URL relatives.

Balise Bold

La balise `<b>` permet de mettre le texte en gras, comme dans l'exemple suivant :

```
text3_txt.htmlText = "He was <b>ready</b> to leave!";
```

Les caractères gras doivent être disponibles dans la police utilisée pour afficher le texte.

Balise Break

La balise `<br>` introduit un saut de ligne dans le champ texte. Le champ de texte doit être multiligne pour utiliser cette balise.

Dans l'exemple suivant, le saut de ligne sépare deux phrases :

```
this.createTextField("text1_txt", 1, 10, 10, 200, 100);
text1_txt.html = true;
text1_txt.multiline = true;
text1_txt.htmlText = "The boy put on his coat.<br />His coat was <font
  color='#FF0033'>red</font> plaid.";
```

Balise Font

La balise `<font>` spécifie une police ou une liste de polices pour l'affichage du texte.

La balise font prend en charge les attributs suivants :

- `color` Seules les valeurs de couleur hexadécimales (`#FFFFFF`) sont prises en charge. Par exemple, le code HTML suivant crée du texte rouge :

  ```
  myText_txt.htmlText = "<font color='#FF0000'>This is red text</font>";
  ```

- `face` Spécifie le nom de la police à utiliser. Comme indiqué dans l'exemple suivant, vous pouvez spécifier des noms de police séparés par des virgules, auquel cas Flash Player sélectionne la première balise disponible :

  ```
  myText_txt.htmlText = "<font face='Times, Times New Roman'>Displays as
    either Times or Times New Roman...</font>";
  ```

  Si la police spécifiée n'est pas installée sur le système informatique de l'utilisateur ou si elle n'est pas intégrée dans le fichier SWF, Flash Player sélectionne une police de remplacement.

  Pour plus d'informations sur l'intégration des polices dans les applications Flash, consultez la section embedFonts (propriété TextField.embedFonts) dans le *Guide de référence du langage ActionScript 2.0* et dans le guide *Utilisation de Flash*.

- `size` Spécifie la taille de la police en pixels, comme indiqué dans l'exemple suivant :

  ```
  myText_txt.htmlText = "<font size='24' color='#0000FF'>This is blue,
    24-point text</font>";
  ```

  Vous pouvez aussi utiliser des tailles de points relatives au lieu d'une taille de pixels, par exemple +2 ou -4.

Balise Image

La balise `<img>` vous permet d'intégrer des fichiers image (JPEG, GIF, PNG), des fichiers SWF et des clips externes à l'intérieur des champs de texte et des occurrences de composant TextArea. Le texte se déroule automatiquement autour des images intégrées dans les champs de texte ou les composants. Pour utiliser cette balise, vous devez définir votre champ de texte dynamique ou de saisie sur le multiligne et le retour à la ligne.

Pour créer un champ de texte multiligne avec retour à la ligne automatique, effectuez l'une des opérations suivantes :

- Dans l'environnement auteur Flash, sélectionnez un champ de texte sur la scène, puis, dans l'inspecteur des propriétés, sélectionnez Multiligne dans le menu contextuel Type de texte.
- Pour un champ de texte créé à l'exécution à l'aide de createTextField (méthode MovieClip.createTextField), définissez les propriétés multiline (propriété TextField.multiline) et multiline (propriété TextField.multiline) de l'occurrence du nouveau champ de texte sur `true`.

La balise `<img>` a un attribut requis, `src`, qui spécifie le chemin vers un fichier image, un fichier SWF ou l'identifiant de liaison d'un symbole de clip dans la bibliothèque. Tous les autres attributs sont facultatifs.

La balise `<img>` prend en charge les attributs suivants :

- `src` Spécifie l'URL vers un fichier image ou SWF, ou l'identifiant de liaison pour un symbole de clip dans la bibliothèque. Cet attribut est requis ; tous les autres attributs sont facultatifs. Les fichiers externes (JPEG, GIF, PNG et SWF) ne s'affichent pas tant qu'ils ne sont pas entièrement téléchargés.
- `id` Spécifie le nom d'une occurrence de clip (créée par Flash Player) contenant le fichier image, SWF ou le clip intégré. Cette fonction s'avère utile pour contrôler le contenu intégré avec ActionScript.
- `width` Largeur de l'image, du fichier SWF ou du clip inséré, en pixels.
- `height` Hauteur de l'image, du fichier SWF ou du clip, en pixels.
- `align` Spécifie l'alignement horizontal de l'image intégrée dans le champ de texte. Les valeur valides sont `left` et `right`. La valeur par défaut est `left`.
- `hspace` Spécifie l'espace horizontal qui entoure l'image là où aucun texte n'apparaît. La valeur par défaut est 8.
- `vspace` Spécifie l'espace vertical qui entoure l'image là où aucun texte n'apparaît. La valeur par défaut est 8.

Pour plus d'informations et d'exemples sur l'utilisation de la balise `<img>`, consultez « Présentation de l'intégration d'images, de fichiers SWF et de clips dans les champs de texte », à la page 437.

Balise Italic

La balise `<i>` affiche le texte qu'elle entoure en italiques, comme dans l'exemple de code suivant :

```
That is very <i>interesting</i>.
```

Cet exemple de code produirait le résultat suivant :

That is very *interesting*.

Des caractères italiques doivent être disponibles dans la police utilisée.

Balise List item

La balise `<li>` place une puce devant le texte qu'elle encadre, comme dans l'exemple de code suivant :

```
Grocery list:
<li>Apples</li>
<li>Oranges</li>
<li>Lemons</li>
```

Cet exemple de code produirait le résultat suivant :

Grocery list :

- Apples
- Oranges
- Lemons

REMARQUE

Les listes ordonnées et non ordonnées (balises `<ol>` et `<ul>`) ne sont pas reconnues par Flash Player de sorte qu'elles ne modifient pas le rendu de votre liste. Tous les éléments de la liste utilisent des puces.

Balise Paragraph

La balise `<p>` crée un paragraphe. Le champ de texte doit être multiligne pour utiliser cette balise.

La balise `<p>` prend en charge les attributs suivants :

- `align` Spécifie l'alignement du texte dans le paragraphe. Les valeurs valides sont `left`, `right`, `justify` et `center`.
- `class` Spécifie une classe de style CSS définie par un objet TextField.StyleSheet. (Pour plus d'informations, voir « Utilisation des classes de style », à la page 420.)

  L'exemple suivant utilise l'attribut `align` pour aligner le texte sur le côté droit d'un champ de texte.

```
this.createTextField("myText_txt", 1, 10, 10, 400, 100);
myText_txt.html = true;
myText_txt.multiline = true;
myText_txt.htmlText = "<p align='right'>This text is aligned on the
  right side of the text field</p>";
```

L'exemple suivant utilise l'attribut `class` pour affecter une classe de style de texte à une balise `<p>`.

```
var myStyleSheet:TextField.StyleSheet = new TextField.StyleSheet();
myStyleSheet.setStyle(".blue", {color:'#99CCFF', fontSize:18});
this.createTextField("test_txt", 10, 0, 0, 300, 100);
test_txt.html = true;
test_txt.styleSheet = myStyleSheet;
test_txt.htmlText = "<p class='blue'>This is some body-styled text.</
  p>.";
```

Balise Span

La balise `<span>` peut uniquement être utilisée avec les styles de texte CSS. (Pour plus d'informations, voir « Formatage de texte avec les feuilles de style CSS », à la page 413.) Elle prend en charge l'attribut suivant :

- `class` Spécifie une classe de style CSS définie par un objet TextField.StyleSheet. Pour plus d'informations sur la création de classes de style, consultez la section « Utilisation des classes de style », à la page 420.

Balise Text format

La balise `<textformat>` permet d'utiliser un sous-ensemble de propriétés de formatage des paragraphes de la classe TextFormat dans les champs de texte HTML, dont l'interlignage, le retrait, les marges et les taquets de tabulation. Vous pouvez associer des balises `<textformat>` aux balises HTML intégrées.

La balise `<textformat>` possède les attributs suivants :

- `blockindent` Spécifie l'indentation d'un bloc, en points. Correspond à `TextFormat.blockIndent`. (Consultez la rubrique blockIndent (propriété TextFormat.blockIndent) dans le *Guide de référence du langage ActionScript 2.0*.)
- `indent` Spécifie l'indentation, de la marge gauche au premier caractère du paragraphe. Correspond à `TextFormat.indent`. Permet d'utiliser des entiers négatifs. (Consultez la rubrique indent (propriété TextFormat.indent) dans le *Guide de référence du langage ActionScript 2.0*.)
- `leading` Spécifie l'espace séparant les lignes (espace vertical). Correspond à `TextFormat.leading`. Permet d'utiliser des entiers négatifs. (Consultez la rubrique leading (propriété TextFormat.leading) dans le *Guide de référence du langage ActionScript 2.0*.)
- `leftmargin` Spécifie la marge gauche du paragraphe, en points. Correspond à `TextFormat.leftMargin`. (Consultez la rubrique leftMargin (propriété TextFormat.leftMargin) dans le *Guide de référence du langage ActionScript 2.0*.)

- `rightmargin` Spécifie la marge droite du paragraphe, en points. Correspond à `TextFormat.rightMargin`. (Consultez la rubrique rightMargin (propriété TextFormat.rightMargin) dans le *Guide de référence du langage ActionScript 2.0*.)
- `tabstops` Spécifie des taquets de tabulation personnalisés, sous forme d'un tableau d'entiers non négatifs. Correspond à `TextFormat.tabStops`. (Consultez la rubrique tabStops (propriété TextFormat.tabStops) dans le *Guide de référence du langage ActionScript 2.0*.)

Le tableau de données suivant présentant des en-têtes de ligne en gras est produit par l'exemple de code illustré dans la procédure ci-après :

Name	Age	Occupation
Rick	33	Détective
AJ	34	Détective

Pour créer un tableau de données formaté en utilisant des taquets de tabulation :

1. Créez un document Flash, puis enregistrez-le sous le nom de **tabstops.fla**.
2. Dans le scénario, sélectionnez la première image sur le Calque 1.
3. Ouvrez le panneau Actions (Fenêtre > Actions) et ajoutez-lui le code suivant :

```
// Créer un nouveau champ de texte
this.createTextField("table_txt", 99, 50, 50, 450, 100);
table_txt.multiline = true;
table_txt.html = true;
// Crée des en-têtes de colonne, formatés en gras et séparés par
// des balises.
var rowHeaders:String = "<b>Name\tAge\tOccupation</b>";

// Crée des lignes contenant des données.
var row_1:String = "Rick\t33\tDetective";
var row_2:String = "AJ\t34\tDetective";

// Définit deux tabulations, à 50 et 100 points.
table_txt.htmlText = "<textformat tabstops='[50,100]'>";
table_txt.htmlText += rowHeaders;
table_txt.htmlText += row_1;
table_txt.htmlText += row_2 ;
table_txt.htmlText += "</textformat>";
```

   L'utilisation de la séquence d'échappement des caractères de tabulation (`\t`) ajoute des tabulations entre chaque colonne dans le tableau. Vous ajoutez le texte à l'aide de l'opérateur +=.

4. Choisissez Contrôle > Tester l'animation pour visualiser le tableau formaté.

Balise Underline

La balise <u> souligne le texte qu'elle entoure, comme dans l'exemple de code suivant :

```
This is <u>underlined</u> text.
```

Ce code produirait le résultat suivant :

This is underlined text.

Présentation des entités HTML prises en charge

Les entités HTML vous aident à afficher certains caractères dans les textes au format HTML, afin qu'ils ne soient pas interprétés comme du HTML. Par exemple, vous utilisez les caractères inférieur à (<) et supérieur à (>) pour encadrer les balises HTML, telles que <img> et . Pour afficher les caractères inférieur à ou supérieur à dans des champs de texte au format HTML dans Flash, vous devez remplacer ces caractères par des entités HTML. L'ActionScript suivant crée un champ de texte au format HTML sur la scène et utilise des entités HTML pour afficher la chaîne « <b> » sans que le texte apparaisse en gras :

```
this.createTextField("my_txt", 10, 100, 100, 100, 19);
my_txt.autoSize = "left";
my_txt.html = true;
my_txt.htmlText = "The &lt;b&gt; tag makes text appear <b>bold</b>.";
```

A l'exécution, l'exemple de code précédent dans Flash affiche le texte suivant sur la scène :

The <b> tag makes text appear **bold**.

Outre les symboles supérieur à et inférieur à, Flash reconnaît également d'autres entités HTML répertoriées dans le tableau suivant.

Entité	Description
`<`	< (Inférieur à)
`>`	> (Supérieur à)
`&`	& (esperluette)
`&quot;`	" (guillemets doubles)
`&apos;`	' (apostrophe, guillemet simple)

Flash prend également en charge les codes de caractères explicites, tels que `&#39;` (esperluette - ASCII) et `&#x0026;` (esperluette - Unicode).

L'ActionScript suivant démontre l'utilisation des codes de caractères ASCII ou Unicode pour intégré un caractère tilde (~) :

```
this.createTextField("my_txt", 10, 100, 100, 100, 19);
my_txt.autoSize = "left";
my_txt.html = true;
my_txt.htmlText = "&#126;"; // tilde (ASCII)
my_txt.htmlText += "\t"
my_txt.htmlText += "&#x007E;"; // tilde (Unicode)
```

Présentation de l'intégration d'images, de fichiers SWF et de clips dans les champs de texte

Dans Flash Player 7 et les versions ultérieures, vous pouvez utiliser la balise `<img>` pour intégrer des fichiers image (JPEG, GIF, PNG), des fichiers SWF et des clips dans les champs de texte dynamiques et de saisie et les occurrences du composant TextArea. (pour obtenir la liste complète des attributs de la balise `<img>`, consultez « Balise Image », à la page 431).

Flash affiche le média intégré dans un champ de texte, à sa taille normale. Pour spécifier les dimensions du média à intégrer, utilisez les attributs `height` et `width` de la balise `<img>`. (Voir la section « Présentation de la spécification des valeurs de hauteur et de largeur », à la page 439.)

En général, une image intégrée dans un champ de texte apparaît sur la ligne qui suit la balise `<img>`. Cependant, lorsque la balise `<img>` est le premier caractère dans le champ de texte, l'image apparaît sur la première ligne du champ de texte.

Intégration de fichiers image et SWF

Pour intégrer un fichier image ou SWF dans un champ de texte, spécifiez le chemin absolu ou relatif qui mène au fichier image (GIF, JPEG ou PNG) ou au fichier SWF dans l'attribut `<sec>` de la balise `img`. Ainsi, l'exemple code suivant insère un fichier GIF situé dans le même répertoire que le fichier SWF (une adresse relative, en ligne ou non).

Intégration d'une image dans un champ de texte :

1. Créez un document Flash, puis enregistrez-le sous **embedding.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
this.createTextField("image1_txt", 10, 50, 50, 450, 150);
image1_txt.html = true;
image1_txt.htmlText = "<p>Here's a picture from my vacation:<img
  src='beach.gif'>";
```

   Le code précédent crée un nouveau champ de texte dynamique sur la scène, active le formatage HTML et ajoute du texte et une image locale au champ de texte.

3. Ajoutez l'ActionScript suivant sous le code ajouté à l'étape précédente :

```
this.createTextField("image2_txt", 20, 50, 200, 400, 150);
image2_txt.html = true;
image2_txt.htmlText = "<p>Here's a picture from my garden:<img
  src='http://www.helpexamples.com/flash/images/image2.jpg'>";
```

   Vous pouvez aussi insérer une image en utilisant une adresse absolue. Le code précédent insère un fichier JPEG situé dans un répertoire qui se trouve sur un serveur. Le fichier SWF contenant ce code peut se trouver sur votre disque dur ou sur un serveur.

4. Enregistrez le document et choisissez Contrôle > Tester l'animation pour tester le document.

   Le champ de texte supérieur devrait contenir une phrase de texte et plus probablement un message d'erreur dans le panneau Sortie indiquant que Flash n'a pas trouvé un fichier nommé beach.gif dans le répertoire courant. Le champ de texte inférieur doit contenir une phrase de texte et l'image d'une fleur qui a été chargée à partir d'un serveur distant.

   Copiez une image GIF vers le même répertoire que le fichier FLA et renommez l'image beach.gif, puis sélectionnez Contrôle > Tester l'animation afin de tester de nouveau le document Flash.

REMARQUE	Lorsque vous utilisez des URL absolues, vous devez veiller à ce que l'URL ait le préfixe `http://`.

Intégration de symboles de clip

Pour intégrer un symbole de clip dans un champ de texte, spécifiez l'identifiant de liaison du symbole pour l'attribut `src` de la balise `<img>` (Pour plus d'informations sur la définition d'un identifiant de liaison, consultez « Association d'un symbole de clip à la scène », à la page 346).

Par exemple, le code suivant insère un symbole de clip avec l'identificateur de liaison `symbol_ID` dans un champ texte dynamique avec le nom d'occurrence `textField_txt`.

Pour intégrer un clip dans un champ de texte :

1. Créez un document Flash, puis enregistrez-le sous le nom de **embeddedmc.fla**.
2. Tracez une nouvelle forme sur la scène ou sélectionnez Fichier > Importer > Importer dans la scène, puis sélectionnez une image mesurant environ 100 pixels de large sur 100 pixels de haut.
3. Convertissez la forme ou l'image importée à l'étape précédente en la sélectionnant sur la scène et en appuyant sur la touche F8 pour ouvrir la boîte de dialogue Convertir en symbole.

4. Définissez le comportement sur Clip et saisissez un nom descriptif pour le symbole. Sélectionnez le carreau supérieur gauche de la grille du point d'alignement, puis cliquez sur Avancé pour passer au mode avancé si ce n'est pas encore fait.
5. Cochez les cases Exportation d'ActionScript et Exporter dans la première image.
6. Saisissez l'identifiant de liaison **img_id** dans le champ de texte Identifiant, puis cliquez sur OK.
7. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
this.createTextField("textField_txt", 10, 0, 0, 300, 200);
textField_txt.html = true;
textField_txt.htmlText = "<p>Here's a movie clip symbol:<img
  src='img_id'>";
```

   Pour qu'un clip intégré soit correctement et entièrement affiché, le point d'alignement de son symbole doit être (0,0).
8. Enregistrez vos modifications dans le document Flash.
9. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

Présentation de la spécification des valeurs de hauteur et de largeur

Si vous spécifiez les attributs `width` et `height` d'une balise `<img>`, un espace est réservé dans le champ de texte pour le fichier image, le fichier SWF ou le clip. Une fois qu'un fichier image ou SWF est complètement téléchargé, il apparaît dans l'espace réservé. Flash modifie la taille du média en fonction des valeurs que vous spécifiez pour `height` et `width`. Vous devez entrer des valeurs pour les attributs `height` et `width` pour pouvoir redimensionner l'image.

Si vous ne spécifiez pas de valeurs pour `height` et `width`, aucun espace n'est réservé au média intégré. Une fois le fichier JPEG ou SWF téléchargé, Flash l'insère dans le champ de texte à sa taille normale et sépare de nouveau le texte autour de lui.

REMARQUE

Si vous chargez de façon dynamique vos images dans un champ texte contenant déjà du texte, la meilleure pratique consiste à spécifier la largeur et la hauteur de l'image d'origine de façon à ce que le texte s'enroule correctement autour de l'espace réservé à l'image.

Contrôle du média intégré avec ActionScript

Flash Player crée un nouveau clip pour chaque balise `<img>` et l'intègre dans l'objet TextField. L'attribut `id` de la balise `<img>` permet d'affecter un nom d'occurrence au clip créé. Cela vous permet de contrôler le clip avec ActionScript.

Le clip créé par Flash est ajouté en tant que clip enfant du champ de texte contenant l'image.

Ainsi, l'exemple suivant intègre un fichier SWF dans un champ de texte.

Pour intégrer un fichier SWF dans un champ de texte :

1. Créez un document Flash.
2. Redimensionnez la scène du document à 100 pixels sur 100 pixels.
3. A l'aide de l'outil Rectangle, dessinez un carré rouge sur la scène.
4. Redimensionnez le carré à 80 pixels sur 80 pixels à l'aide de l'inspecteur des propriétés, puis déplacez la forme vers le centre de la scène.
5. Sélectionnez l'Image 20 du scénario, puis appuyez sur la touche F7 (Windows ou Macintosh) pour insérer une nouvelle image clé vierge.
6. Utilisez l'outil Ovale pour tracer un cercle bleu sur la scène dans l'Image 20.
7. Redimensionnez le cercle à 80 pixels sur 80 pixels à l'aide de l'inspecteur des propriétés, puis déplacez-le vers le centre de la scène.
8. Cliquez sur une image vierge entre les Images 1 et 20, puis définissez le type d'interpolation sur Forme dans l'inspecteur des propriétés.
9. Enregistrez le document courant sous **animation.fla**.
10. Choisissez Contrôle > Tester l'animation pour prévisualiser l'animation.

    Le fichier SWF est créé dans le même répertoire que le fichier FLA. Pour que cet exercice fonctionne correctement, le fichier SWF doit se générer pour vous permettre de le charger dans un fichier FLA distinct.
11. Créez un fichier FLA, puis enregistrez-le sous le nom de **animationholder.fla**.

    Enregistrez le fichier dans le même dossier que le fichier animation.fla que vous avez créé précédemment.
12. Ajoutez le code ActionScript suivant à l'image1 du scénario principal :

```
this.createTextField("textField_txt", 10, 0, 0, 300, 200);
textField_txt.html = true;
textField_txt.htmlText = "Here's an interesting animation: <img
  src='animation.swf' id='animation_mc'>";
```

    Dans ce cas, le chemin entièrement qualifié vers le nouveau clip créé est `textField_txt.animation_mc`.

13. Enregistrez les modifications apportées au document Flash, puis sélectionnez Contrôle > Tester l'animation pour prévisualiser l'animation dans le champ de texte.

Pour contrôler le fichier SWF lorsqu'il est lu dans un fichier texte, effectuez l'exercice suivant.

Pour contrôler un fichier SWF qui est lu dans un champ de texte :

1. Suivez les étapes de la première procédure sous « Contrôle du média intégré avec ActionScript », à la page 440.
2. Créez une occurrence de bouton sur la scène et attribuez-lui le nom d'occurrence **stop_btn** dans l'inspecteur des propriétés.
3. Ajoutez le code ActionScript suivant sous le code existant dans l'image 1 du scénario principal :

```
stop_btn.onRelease = function() {
  textField_txt.animation_mc.stop();
};
```

4. Sélectionnez Contrôle > Tester l'animation pour tester l'application.

   Désormais, chaque fois que vous cliquerez sur l'occurrence de bouton `stop_btn`, le scénario de l'animation imbriqué dans le champ de texte s'interrompra.

Pour plus d'informations sur la transformation de vos médias imbriqués en hyperliens, consultez la section « Présentation de la transformation de médias imbriqués en liens hypertexte », à la page 441.

Présentation de la transformation de médias imbriqués en liens hypertexte

Pour créer un lien hypertexte à partir d'un fichier image, d'un fichier SWF ou d'un clip intégré, incluez la balise `<img>` dans une balise `<a>` :

```
textField_txt.htmlText = "Click the image to return home<a
  href='home.htm'><img src='home.jpg'></a>";
```

Lorsque le pointeur de la souris survole une image, un fichier SWF ou un clip que vous avez placé entre des balises `<a>`, il prend la forme d'une « main qui pointe son index », à l'instar des hyperliens standard. L'interactivité, telle que les clics de souris et la pression sur les touches du clavier, n'est pas enregistrée dans les fichiers SWF et les clips placés entre les balises `<a>`.

Pour plus d'informations sur l'imbrication de médias, consultez la section « Présentation de la transformation de médias imbriqués en liens hypertexte », à la page 441.

Exemple : Création de texte défilant

Plusieurs méthodes sont à votre disposition pour créer un texte de défilement dans Flash. Pour faire défiler des champs de texte dynamique et de saisie, vous pouvez sélectionner l'option Défilant dans le menu Texte ou le menu contextuel ou encore double-cliquer sur la poignée du champ de texte tout en maintenant la touche Maj enfoncée.

Vous pouvez utiliser les propriétés `scroll` et `maxscroll` de l'objet TextField pour contrôler le défilement vertical et les propriétés `hscroll` et `maxhscroll` pour contrôler le défilement horizontal d'un champ de texte. Les propriétés `scroll` et `hscroll` spécifient respectivement les positions de défilement vertical et horizontal ; vous pouvez lire et définir ces propriétés. Les propriétés `maxscroll` et `maxhscroll` spécifient les positions verticales et horizontales maximales, respectivement ; vous pouvez uniquement lire ces propriétés.

Le composant TextArea offre un moyen aisé de créer un champ de texte défilant avec un minimum de programmation. Pour plus d'informations, consultez la section relative « composant TextArea » dans la *Référence du langage des composants ActionScript 2.0*.

Pour créer un champ de texte dynamique à défilement :

Effectuez l'une des opérations suivantes :

- Double-cliquez sur la poignée du champ texte dynamique tout en maintenant la touche maj enfoncée.
- Sélectionnez le champ texte dynamique avec l'outil Sélection et choisissez Texte > Défilant.
- Sélectionnez le champ texte dynamique avec l'outil Sélection. Cliquez avec le bouton droit de la souris (Windows) ou tout en appuyant sur la touche Contrôle (Macintosh) sur le champ texte dynamique et sélectionnez Texte > Défilant.

Pour utiliser la propriété scroll afin de créer du texte défilant :

1. Effectuez l'une des opérations suivantes :
   - Sélectionnez l'outil Texte et tracez un champ de texte sur la scène. Affectez le nom d'occurrence **textField_txt** au champ de texte dans l'inspecteur des propriétés.
   - Utilisez ActionScript pour créer un champ de texte dynamiquement à l'aide de la méthode `MovieClip.createTextField()`. Affectez le nom d'occurrence **textField_txt** au champ de texte en tant que paramètre de la méthode.

REMARQUE	Si vous ne chargez *pas* le texte de façon dynamique dans le fichier SWF, sélectionnez Texte > Défilant dans le menu principal.

2. Créez un bouton Vers le haut et un bouton Vers le bas ou sélectionnez Fenêtre > Bibliothèques communes > Boutons, puis faites glisser vos boutons sur la scène.

   Vous utiliserez ces boutons pour faire défiler le texte vers le haut et vers le bas.

3. Sélectionnez le bouton Vers le bas sur la scène et tapez **down_btn** dans le champ Nom de l'occurrence.

4. Sélectionnez le bouton Vers le haut sur la scène et tapez **up_btn** dans le champ Nom de l'occurrence.

5. Sélectionnez Image 1 dans le scénario et, dans le panneau Actions Fenêtre > Actions), saisissez le code suivant pour faire défiler le texte vers le bas dans le champ de texte :

```
down_btn.onPress = function() {
  textField_txt.scroll += 1;
};
```

6. Suite à l'ActionScript de l'étape 5, saisissez le code suivant pour faire défiler le texte vers le haut :

```
up_btn.onPress = function() {
  textField_txt.scroll -= 1;
};
```

   Tout texte se chargeant dans le champ `textField_txt` peut défiler à l'aide des boutons Vers le haut et Vers le bas.

Présentation des chaînes et de la classe String

En programmation, une chaîne est une série ordonnée de caractères. Les chaînes sont souvent utilisées dans les documents Flash et les fichiers de classe pour afficher du texte dans des applications, par exemple dans des champs de texte. De même, vous pouvez stocker des valeurs sous forme de chaînes exploitables dans une application pour diverses raisons. Vous pouvez créer les chaînes directement dans votre code ActionScript en plaçant les caractères de données entre guillemets. Pour plus d'informations sur la création de chaînes, consultez la section « Création de chaînes », à la page 452. Pour plus d'informations sur l'utilisation des champs de texte, veuillez consulter la rubrique « Utilisation de la classe TextField », à la page 370.

Vous pouvez associer chaque caractère à un code de caractère spécifique, que vous pouvez même éventuellement utiliser pour afficher du texte. Par exemple, le caractère « A » est représenté par le caractère Unicode 0041, ou 65 en ASCII (American Standard Code for Information Interchange). Pour plus d'informations sur les codes de caractères et les tableaux de codes, consultez le site www.unicode.org/charts. Comme vous pouvez le constater, la manière dont vous représentez les chaînes dans un document Flash dépend beaucoup du jeu de caractères choisi et de la manière dont vous encodez les caractères.

Le codage de caractères fait référence au code, ou à la méthode, utilisé(e) pour représenter un jeu de caractères d'une langue en codes représentatifs, tels que des valeurs numériques. Le *code de caractères* (tel que mentionné au paragraphe précédent) est le tableau des valeurs mappées (par exemple, le tableau ASCII, dans lequel le A est égal à 65). La méthode de codage le déchiffre dans un programme informatique.

Par exemple, chaque lettre de la langue anglaise correspond à un code numérique représentatif dans un codage de caractères. L'ASCII encode chaque lettre, chaque chiffre ainsi que certains symboles en versions binaires 7 bits de chaque entier. L'ASCII est un jeu de caractères comprenant 95 caractères imprimables et de nombreux caractères de contrôle ; l'ASCII est utilisé par les ordinateurs pour représenter le texte.

Comme le standard ASCII, le codage *Unicode* est un autre moyen d'associer un code à chaque lettre de l'alphabet. Dans la mesure où l'ASCII ne prend pas en charge les jeux de caractères volumineux, tels que le Chinois, le Standard Unicode est utile pour encoder les langues. Unicode est le standard utilisé pour les jeux de caractères et peut représenter n'importe quelle langue. Ce standard est destiné à faciliter le développement d'applications en plusieurs langues. Le code de caractère désigne le caractère qu'il représente et le standard tente de fournir un moyen universel pour coder tous les caractères constituant une langue. Les chaînes peuvent alors être affichées n'importe où, quel que soit le système informatique, la plate-forme ou le logiciel. L'application impliquée (telle que Flash ou un navigateur Web) doit alors être capable d'afficher les glyphes (la représentation graphique) de chaque caractère.

Au fil des années, le nombre de caractères pris en charge par Unicode s'est développé pour intégrer la prise en charge de langues plus nombreuses (et plus volumineuses). Les encodages de caractères sont appelés UTF (Unicode Transformation Format) et UCS (Universal Character Set), et comprennent les codes UTF-8, UTF-16 et UTF-32. Les nombres suivant l'encodage UTF représentent le nombre de bits présents dans une unité et les nombres suivant l'encodage UCS représentent des octets.

- UTF-8 est actuellement le codage standard pour les échanges de texte, tels que les systèmes de messagerie en ligne. L'UTF est un système à 8 bits.
- L'UTF-16 est utilisé couramment pour le traitement interne.

La longueur des chaînes peut varier dans vos applications. Vous pouvez déterminer cette longueur, même si elle peut varier, en fonction de la langue utilisée. De même, vous pouvez rencontrer un caractère de terminaison (null) à la fin d'une chaîne, qui n'a pas de valeur. Ce caractère de terminaison n'est pas un véritable caractère, mais vous pouvez l'utiliser pour localiser la fin d'une chaîne. Par exemple, si vous utilisez des connexions socket, vous pouvez rechercher le caractère de terminaison pour localiser la fin d'une chaîne (dans un logiciel de discussion en ligne par exemple).

Pour un exemple de fichier source, strings.fla, qui vous décrit comment construire un traitement de texte simple qui compare et recouvre des sélections de chaînes et de sous-chaînes, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Strings afin d'accéder à l'exemple.

Pour plus d'informations sur les chaînes et sur la classe String, consultez les rubriques suivantes :

panneau

Le panneau Chaînes permet de créer et mettre à jour du contenu multilingue. Vous pouvez spécifier un contenu pour les champs de texte recouvrant plusieurs langues, et demander à Flash de déterminer automatiquement le contenu qui doit apparaître dans une certaine langue en fonction de la langue de l'ordinateur qui exécute Flash Player.

Pour obtenir des informations générales sur le panneau Chaînes et pour savoir comment les utiliser dans vos applications, consultez les rubriques suivantes dans *Utilisation de Flash* :

- « Programmation de texte multilingue »
- « Présentation du panneau Chaînes »
- « Traduction de texte dans le panneau Chaînes ou dans un fichier XML »
- « Importation d'un fichier XML dans le panneau Chaînes »

Vous pouvez utiliser la classe Locale pour contrôler l'affichage du texte multilingue. Pour plus d'informations, consultez les sections « Utilisation de la classe Locale », à la page 446 et `Locale (mx.lang.Locale)` dans le *Guide de référence du langage ActionScript 2.0*.

Utilisation de la classe Locale

La classe Locale (mx.lang.Locale) vous permet de contrôler l'affichage du texte multilingue dans une application Flash à l'exécution. Le panneau Chaînes vous permet d'utiliser les identifiants de chaînes au lieu de leur texte littéral dans des champs de texte dynamiques, ce qui vous permet de créer un fichier SWF affichant le texte chargé à partir d'un fichier XML spécifique à la langue. Vous pouvez utiliser les méthodes suivantes pour afficher les chaînes spécifiques à la langue figurant dans les fichiers XlIFF (XML Localization Interchange File Format).

Automatiquement à l'exécution Flash Player remplace les identifiants de chaînes par des chaînes provenant du fichier XML, lequel correspond au code de langue par défaut du système qui est retourné par `language (propriété capabilities.language)`.

Manuellement à l'aide du langage de la scène Les identifiants de chaînes sont remplacés par des chaînes lorsque le fichier SWF est compilé et ne peuvent pas être modifiés par Flash Player.

Utilisation d'ActionScript à l'exécution Vous pouvez contrôler le remplacement des identifiants de chaînes à l'aide d'ActionScript, lequel est contrôlé à l'exécution. Cette option vous permet de contrôler le moment et la langue du remplacement de l'identifiant de chaîne.

Vous pouvez utiliser les propriétés et les méthodes de la classe Locale lorsque vous souhaitez remplacer les identifiants de chaîne à l'aide d'ActionScript pour contrôler l'application lorsqu'elle est exécutée dans Flash Player. Pour obtenir une démonstration de l'utilisation de la classe Locale, respecez la procédure suivante :

Pour utiliser la classe Locale pour créer des sites multilingues :

1. Créez un document Flash, puis enregistrez-le sous le nom de **locale.fla**.
2. Ouvrez le panneau Chaînes (Fenêtre > Autres panneaux > Chaînes) et cliquez sur Paramètres.
3. Sélectionnez deux langues, en (anglais) et fr (français), puis cliquez sur Ajouter pour ajouter les langues au panneau Langues actives.
4. Sélectionnez l'option Via ActionScript à l'exécution, définissez la langue d'exécution par défaut sur Français, puis cliquez sur OK.
5. Faites glisser un composant ComboBox de du dossier User Interface du panneau Composants (Fenêtre > Composants) sur la scène et attribuez-lui le nom d'occurrence **lang_cb**.
6. Créez un champ de texte dynamique sur la scène à l'aide de l'outil Texte et donnez au champ de texte le nom d'occurrence **greeting_txt**.

7. Après avoir sélectionné le champ de texte sur la scène, tapez l'identifiant de chaîne **greeting** dans le champ ID du panneau Chaînes, puis cliquez sur Appliquer.

   Vous remarquerez que Flash convertit la chaîne d'accueil en IDS_GREETING.

8. Dans la grille du panneau Chaînes, tapez la chaîne **hello** dans la colonne en.

9. Tapez la chaîne **bonjour** dans la colonne fr.

   Vous utilisez ces chaînes lorsque vous utilisez la liste déroulante `lang_cb` pour modifier la langue de la scène.

10. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
import mx.lang.Locale;
Locale.setLoadCallback(localeListener);
lang_cb.dataProvider = Locale.languageCodeArray.sort();
lang_cb.addEventListener("change", langListener);
greeting_txt.autoSize = "left";
Locale.loadLanguageXML(lang_cb.value);

function langListener(eventObj:Object):Void {
  Locale.loadLanguageXML(eventObj.target.value);
}
function localeListener(success:Boolean):Void {
  if (success) {
    greeting_txt.text = Locale.loadString("IDS_GREETING");
  } else {
    greeting_txt.text = "unable to load language XML file.";
  }
}
```

   L'ActionScript précédent est divisé en deux sections. La première section de code importe la classe Local et spécifie un écouteur de rappel qui sera appelé chaque fois que le chargement d'un fichier XML de langue sera terminé. La liste déroulante `lang_cb` est ensuite remplie par un tableau trié des langues disponibles. Chaque fois que la valeur `lang_cb` est modifiée, le diffuseur d'événements de Flash déclenche la fonction `langListener()`, laquelle charge le fichier XML dans la langue spécifiée. La deuxième section de code définit deux fonctions, `langListener()` et `localeListener()`.
   La première fonction, `langListener()`, est appelée chaque fois que la valeur de la liste déroulante `lang_cb` est modifiée par l'utilisateur. La deuxième fonction, `localeListener()`, est appelée chaque fois que le chargement du fichier XML d'une langue est terminé. Cette fonction vérifie que le chargement a réussi et, si tel est le cas, définit la propriété `text`de l'occurrence `greeting_txt` sur l'accueil dans la langue sélectionnée.

11. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

CONSEIL	Le fichier XML que vous utilisez doit utiliser le format XLIFF (XML Localization Interchange File Format).

ATTENTION	La classe Locale est différente des autres classes dans le Guide de référence du langage ActionScript 2.0, dans la mesure où elle ne fait pas partie de Flash Player. Dans la mesure où cette classe est installée dans le chemin de classe Auteur de Flash, elle est automatiquement compilée dans vos fichiers SWF. L'utilisation de la classe Locale augmente légèrement la taille du fichier SWF, car la classe doit être compilée dans le fichier SWF.

Pour plus d'informations, consultez la section `Locale (mx.lang.Locale)` dans le *Guide de référence du langage ActionScript 2.0*.

Utilisation d'un IME (input method editor)

L'IME (input method editor) permet aux utilisateurs de taper des caractères de texte non-ASCII dans des langues asiatiques telles que le Chinois, le Japonais et le Coréen. La classe IME dans ActionScript vous permet de manipuler directement l'IME du système d'exploitation dans l'application Flash Player qui s'exécute sur un ordinateur client.

A l'aide d'ActionScript, vous pouvez déterminer les éléments suivants :

- Si un IME est installé sur l'ordinateur de l'utilisateur.
- Si l'IME est activé ou désactivé sur l'ordinateur de l'utilisateur.
- Quel mode de conversion est utilisé par l'IME courant.

La classe d'IME peut déterminer le mode de conversion utilisé par l'IME courant : par exemple, si l'IME japonais est actif, vous pouvez déterminer si le mode de conversion est Hiragana, Katakana, etc. à l'aide de la méthode `System.IME.getConversionMode()` . Vous pouvez le définir à l'aide de la méthode `System.IME.setConversionMode()` .

REMARQUE	Actuellement, vous ne pouvez pas définir *quel* IME est actif (éventuellement) ni le modifier (par exemple, remplacer l'anglais par le japonais ou le coréen par le chinois).

Vous pouvez aussi désactiver ou activer l'IME à l'aide de votre application à l'exécution et effectuer d'autres fonctions, selon le système d'exploitation de l'utilisateur. Vous pouvez vérifier si un système est équipé d'un IME à l'aide de la propriété `System.capabilities.hasIME`. L'exemple suivant indique comment déterminer si l'utilisateur dispose d'un IME installé et actif.

Pour déterminer si l'utilisateur dispose d'un IME installé et actif :

1. Créez un document Flash, puis enregistrez-le sous le nom de **ime.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
if (System.capabilities.hasIME) {
  if (System.IME.getEnabled()) {
    trace("You have an IME installed and enabled.");
  } else {
    trace("You have an IME installed but not enabled.");
  }
} else {
  trace("Please install an IME and try again.");
}
```

   Le code précédent détermine d'abord si le système courant est équipé d'un IME. Si un IME est installé, Flash regarde s'il est actuellement activé.

3. Sélectionnez Contrôle > Tester l'animation pour tester le document.

   Un pessage apparaît dans le panneau Sortie, indiquant si un IME est installé et actuellement actif.

Vous pouvez aussi utiliser la classe IME pour activer et désactiver l'IME dans Flash à l'exécution. L'exemple suivant requiert qu'un IME soit installé sur votre système. Pour plus d'informations sur l'installation d'un IME sur votre plate-forme spécifique, consultez les liens suivants :

- www.microsoft.com/globaldev/default.mspx
- http://developer.apple.com/documentation/
- http://java.sun.com

Vous pouvez activer et désactiver un IME pendant que le fichier SWF s'exécute, comme indiqué dans l'exemple suivant.

Pour activer et désactiver un IME à l'exécution :

1. Créez un document Flash, puis enregistrez-le sous le nom de **ime2.fla**.
2. Créez deux occurrences de symboles de boutons sur la scène et donnez-leur les noms d'occurrences **enable_btn** et **disable_btn**.

3. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
checkIME();

var my_fmt:TextFormat = new TextFormat();
my_fmt.font = "_sans";

this.createTextField("ime_txt", 10, 100, 10, 320, 240);
ime_txt.border = true;
ime_txt.multiline = true;
ime_txt.setNewTextFormat(my_fmt);
ime_txt.type = "input";
ime_txt.wordWrap = true;

enable_btn.onRelease = function() {
  System.IME.setEnabled(true);
};
disable_btn.onRelease = function() {
  System.IME.setEnabled(false);
};

function checkIME():Boolean {
  if (System.capabilities.hasIME) {
    if (System.IME.getEnabled()) {
      trace("You have an IME installed and enabled.");
      return true;
    } else {
      trace("You have an IME installed but not enabled.");
      return false;
    }
  } else {
    trace("Please install an IME and try again.");
    return false;
  }
}
```

Ce code se compose de cinq sections différentes. La première section appelle la méthode `checkIME()`, qui affiche un message dans le panneau de sortie si un IME est installé ou activé sur le système. La deuxième section définit un objet format de texte personnalisé, qui définit la police sur `_sans`. La troisième section crée un champ de texte de saisie et applique le format de texte personnalisé. La quatrième section crée des gestionnaires d'événements pour les occurrences `enable_btn` et `disable_btn` créées lors d'une étape précédente. La cinquième et dernière section de code définit la fonction personnalisée `checkIME()`, qui vérifie si le système courant est équipé d'un IME et, si tel est le cas, si cet IME est actif ou non.

4. Enregistrez le fichier FLA et choisissez Contrôle > Tester l'animation pour tester le document.

REMARQUE	Cet exemple requiert qu'un IME soit installé sur votre système. Pour plus d'informations sur l'installation d'un IME, consultez les liens mentionnés plus haut.

Tapez du texte dans le champ de texte de saisie sur la scène. Commutez votre IME sur une autre langue et tapez de nouveau dans le champ de texte de saisie. Flash Player entre des caractères à l'aide du nouvel IME. Si vous cliquez sur le bouton `disable_btn` sur la scène, Flash revient à l'utilisation de la langue précédente et ignore les paramètres d'IME courants.

Pour plus d'informations sur `System.capabilities.hasIME`, consultez la section `hasIME (propriété capabilities.hasIME)` dans le *Guide de référence du langage ActionScript 2.0*.

Présentation de la classe String

Dans le langage ActionScript de base, une chaîne est également une classe et un type de données. Le type de données String représente une séquence de caractères 16 bits qui peuvent comprendre des lettres, des chiffres et des caractères de ponctuation. Les chaînes sont stockées sous forme de caractères Unicode, au format UTF-16. Toute opération effectuée sur la valeur d'une chaîne renvoie une nouvelle occurrence de la chaîne. La valeur par défaut d'une variable déclarée avec ce type de données est `null`.

Pour plus d'informations sur les chaînes, les données et les valeurs, consultez le Chapitre 3, « Données et types de données ».

La classe String contient des méthodes qui vous permettent de manipuler des chaînes de texte. Les chaînes sont importantes pour travailler avec la plupart des objets, et les méthodes décrites dans ce chapitre sont très utiles pour manipuler les chaînes utilisées dans la plupart des objets, tels que les occurrences TextField, XML, ContextMenu et FileReference.

La classe String est une enveloppe pour le type de données de base String et fournit des méthodes et des propriétés qui vous permettent de manipuler les valeurs de base des chaînes. Vous pouvez convertir la valeur de tout objet en chaîne à l'aide de la fonction `String()`. Toutes les méthodes de la classe String, à l'exception de `concat()`, `fromCharCode()`, `slice()` et `substr()`, sont génériques, ce qui signifie que les méthodes appellent la fonction `toString()` avant d'effectuer leurs opérations, et que vous pouvez utiliser ces méthodes avec d'autres objets non String.

Dans la mesure où tous les index des chaînes ont une base de zéro, l'index du dernier caractère de toute chaîne `myStr` est `myStr.length - 1`.

Pour un exemple de fichier source, strings.fla, qui vous décrit comment construire un traitement de texte simple qui compare et recouvre des sélections de chaînes et de sous-chaînes, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Strings afin d'accéder à l'exemple.

Création de chaînes

Vous pouvez appeler n'importe quelle méthode de la classe String à l'aide de la méthode de constructeur `new String()` ou en utilisant une valeur littérale pour la chaîne. Si vous spécifiez une chaîne littérale, l'interprète d'ActionScript la convertit automatiquement en un objet String temporaire, appelle la méthode et élimine l'objet String temporaire. Vous pouvez aussi utiliser la propriété `String.length` avec une chaîne littérale.

Ne confondez pas un *littéral* de chaîne avec un *objet* String. Pour plus d'informations sur la création de littéraux de chaîne et sur l'objet String, consultez le Chapitre 4, « Présentation des littéraux », à la page 99.

Dans l'exemple suivant, la ligne de code crée la chaîne littérale `firstStr`. Pour déclarer un littéral de chaîne, utilisez les guillemets simples droits (') ou les guillemets doubles droits (") comme délimiteurs.

Pour créer et utiliser des chaînes :

1. Créez un document Flash, puis enregistrez-le sous le nom de **strings.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
var firstStr:String = "foo";
var secondStr:String = new String("foo");
trace(firstStr == secondStr); // true
var thirdStr:String;
trace(thirdStr); // non défini
```

   Ce code définit trois objets String, un utilisant un littéral de chaîne, un autre utilisant l'opérateur `new`, et le dernier sans valeur initiale. Les chaînes peuvent être comparées à l'aide de l'opérateur d'égalité (==), comme indiqué dans la troisième ligne de code. En ce qui concerne les variables, vous spécifiez le type de données uniquement lorsque la variable est en cours de définition.
3. Sélectionnez Contrôle > Tester l'animation pour tester le document.

Utilisez toujours des littéraux de chaîne, sauf si vous devez spécifiquement utiliser un objet String. Pour plus d'informations sur la création de littéraux de chaîne et sur l'objet String, consultez le Chapitre 4, « Présentation des littéraux », à la page 99.

Pour utiliser les guillemets simples droits (') et les guillemets doubles droits (") en tant que délimiteurs dans un littéral de chaîne, utilisez la barre oblique inversée (\) comme caractère d'échappement. Les deux chaînes suivantes sont équivalentes :

```
var firstStr:String = "That's \"fine\"";
var secondStr:String = 'That\'s "fine"';
```

Pour plus d'informations sur l'utilisation de la barre oblique inversée dans les chaînes, consultez la section « Présentation du caractère d'échappement », à la page 453.

N'oubliez pas que vous ne pouvez pas utiliser les « guillemets recourbés » ni les « guillemets spéciaux » dans votre code ActionScript ; ils sont différents des guillemets droits (') et (") que vous utilisez dans votre code. Lorsque vous collez du texte à partir d'une autre source dans ActionScript, par exemple depuis le Web ou un document Word, veillez à utiliser des guillemets droits comme délimiteurs.

Pour un exemple de fichier source, strings.fla, qui vous décrit comment construire un traitement de texte simple qui compare et recouvre des sélections de chaînes et de sous-chaînes, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Strings afin d'accéder à l'exemple.

Présentation du caractère d'échappement

Vous pouvez utiliser le caractère d'échappement (\) pour définir d'autres caractères dans des littéraux de chaîne.

Séquence d'échappement	Description
`\b`	Le caractère de retour arrière.
`\f`	Le caractère de changement de page.
`\n`	Le caractère de changement de ligne.
`\r`	Le retour de chariot.
`\t`	Le caractère de tabulation.
`\u`*nnnn*	Le caractère Unicode dont le code est spécifié par le nombre hexadécimal *nnnn*. Par exemple, `\u263a` est le caractère sourire.
`\x`*nn*	Le caractère ASCII dont le code est spécifié par le nombre hexadécimal *nn*.

Séquence d'échappement	Description
\'	Un guillemet simple.
\"	Un guillemet double.
\\	Une barre oblique inversée.

Pour plus d'informations sur les chaînes littérales, consultez le Chapitre 4, « Présentation des littéraux », à la page 99 et la rubrique « Création de chaînes », à la page 452.

Analyse et comparaison des caractères dans les chaînes

Chaque caractère d'une chaîne possède une position d'index dans la chaîne (un entier). La position d'index du premier caractère est 0. Par exemple, dans la chaîne `yellow`, le caractère `y` occupe la position 0 et le caractère `w` occupe la position 5.

Chaque chaîne possède une propriété `length` correspondant au nombre de caractères qu'elle contient :

```
var companyStr:String = "adobe";
trace(companyStr.length); // 10
```

Une chaîne vide et une chaîne nulle présentent une longueur de zéro :

```
var firstStr:String = new String();
trace(firstStr.length); // 0

var secondStr:String = "";
trace(secondStr.length); // 0
```

Si une chaîne ne contient aucune valeur, sa longueur est indéfinie :

```
var thirdStr:String;
trace(thirdStr.length); // non défini
```

AVERTISSEMENT

Si vogtre chaîne contient un caractère d'octet nul (\0), la valeur de la chaîne sera tronquée.

Vous pouvez également définir une chaîne à l'aide de codes de caractère. Pour plus d'informations sur les codes et l'encodage des caractères, consultez la section « Présentation des chaînes et de la classe String », à la page 443.

L'exemple suivant crée une variable nommée `myStr` et définit la valeur de la chaîne en fonction des valeurs ASCII communiquées à la méthode `String.fromCharCode()` :

```
var myStr:String =
  String.fromCharCode(104,101,108,108,111,32,119,111,114,108,100,33);
trace(myStr); // hello world!
```

Chaque nombre figurant dans la méthode `fromCharCode()` dans le code précédent représente un seul caractère. Par exemple, la valeur ASCII de 104 correspond à un *h* minuscule, alors que la valeur ASCII 32 représente le caractère d'échappement.

Vous pouvez aussi utiliser la méthode `String.fromCharCode()` pour convertir les valeurs Unicode, même si la valeur Unicode doit être convertie de valeurs hexadécimales en valeurs décimales, comme indiqué dans l'ActionScript suivant :

```
// Unicode 0068 == "h"
var letter:Number = Number(new Number(0x0068).toString(10));
trace(String.fromCharCode(letter)); // h
```

Vous pouvez examiner les caractères situés à différentes positions dans une chaîne, comme dans l'exemple suivant.

Pour contourner une chaîne :

1. Créez un document Flash.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

   ```
   var myStr:String = "hello world!";
   for (var i:Number = 0; i < myStr.length; i++) {
     trace(myStr.charAt(i));
   }
   ```

3. Choisissez Contrôle > Tester l'animation pour prévisualiser votre document Flash. Chaque caractère doit être tracé dans le panneau Sortie sur une ligne distincte.
4. Modifiez le code ActionScript existant afin qu'il trace la valeur ASCII correspondant à chaque caractère :

   ```
   var myStr:String = "hello world!";
   for (var i:Number = 0; i < myStr.length; i++) {
     trace(myStr.charAt(i) + " - ASCII=" + myStr.charCodeAt(i));
   }
   ```

5. Enregistrez le document Flash courant et choisissez Contrôle > Tester l'animation pour prévisualiser le fichier SWF.

   Lorsque vous exécutez ce code, l'affichage suivant apparaît dans le panneau Sortie :

```
h - ASCII=104
e - ASCII=101
l - ASCII=108
l - ASCII=108
o - ASCII=111
  - ASCII=32
w - ASCII=119
o - ASCII=111
r - ASCII=114
l - ASCII=108
d - ASCII=100
! - ASCII=33
```

CONSEIL Vous pouvez aussi diviser une chaîne en un tableau de caractères à l'aide de la méthode String.split() et en saisissant une `chaîne vide ("")` en tant que délimiteur ; par exemple, `var charArray:Array = myStr.split("");`

Vous pouvez utiliser les opérateurs pour comparer les chaînes. Pour plus d'informations sur l'utilisation des opérateurs avec les chaînes, consultez « Utilisation d'opérateurs avec des chaînes », à la page 151.

Vous pouvez utiliser ces opérateurs avec des instructions conditionnelles, telles que `if` et `while`. L'exemple suivant fait une comparaison à l'aide d'opérateurs et de chaînes.

Pour comparer deux chaînes :

1. Créez un document Flash, puis enregistrez-le sous le nom de **comparestr.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
var str1:String = "Apple";
var str2:String = "apple";
if (str1 < str2) {
  trace("Uppercase letters sort first.");
}
```

3. Enregistrez le document Flash et choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

Vous pouvez utiliser les opérateurs d'égalité (==) et d'inégalité (!=) pour comparer des chaînes avec d'autres types d'objets, comme indiqué dans l'exemple suivant.

Pour comparer les chaînes à d'autres types de données :

1. Créez un document Flash, puis enregistrez-le sous le nom de **comparenum.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
var myStr:String = "4";
var total:Number = 4;
if (myStr == total) {
  trace("Types are converted.");
}
```

3. Enregistrez le document Flash et choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

   Lorsque vous comparez deux types de données différents (par exemple, des chaînes et des nombres), Flash tente de convertir les types de données pour permettre une comparaison.

Utilisez les opérateurs d'égalité stricte (===) et d'inégalité stricte (!==) pour vérifier que les deux objets comparés sont du même type. Les exemples suivants utilisent des opérateurs de comparaison stricte pour s'assurer que Flash ne tente pas de convertir les types de données en essayant de comparer les valeurs.

Pour forcer des comparaisons strictes de types de données :

1. Créez un document Flash, puis enregistrez-le sous le nom de **comparestrict.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
var str1:String = "4";
var str2:String = "5";
var total:Number = 4;
if (str1 !== total) {
  trace("Types are not converted.");
}
if (str1 !== str2) {
  trace("Same type, but the strings don't match.");
}
```

3. Enregistrez le document Flash et sélectionnez Contrôle > Tester l'animation.

Pour plus d'informations sur l'utilisation des opérateurs avec les chaînes, consultez « Utilisation d'opérateurs avec des chaînes », à la page 151.

Pour un exemple de fichier source, strings.fla, qui vous décrit comment construire un traitement de texte simple qui compare et recouvre des sélections de chaînes et de sous-chaînes, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Strings afin d'accéder à l'exemple.

Conversion et concaténation de chaînes

Vous pouvez utiliser la méthode `toString()` pour convertir de nombreux objets en chaînes. La plupart des objets intégrés disposent d'une méthode `toString()` pour ce faire :

```
var n:Number = 0.470;
trace(typeof(n.toString())); // string
```

Lorsque vous utilisez l'opérateur d'addition (+) avec une combinaison d'occurrences de chaînes et d'autres objets, vous n'avez pas à utiliser la méthode `toString()`. Pour plus de détails sur la concaténation, consultez la deuxième procédure de cette section.

Les méthodes `toLowerCase()` et `toUpperCase()` convertissent les caractères alphabétiques des chaînes en majuscules et minuscules, respectivement. L'exemple suivant démontre la conversion d'une chaîne de caractères minuscules en caractères majuscules.

Pour convertir une chaîne de caractères minuscules en majuscules :

1. Créez un document Flash, puis enregistrez-le sous le nom de **convert.fla**.
2. Tapez le code suivant dans l'image 1 du scénario :

```
var myStr:String = "Dr. Bob Roberts, #9.";
trace(myStr.toLowerCase()); // dr. bob roberts, #9.
trace(myStr.toUpperCase()); // DR. BOB ROBERTS, #9.
trace(myStr); // Dr. Bob Roberts, #9.
```

3. Enregistrez le document Flash et choisissez Contrôle > Tester l'animation.

REMARQUE

Une fois que ces méthodes sont exécutées, la chaîne source reste inchangée. Pour transformer la chaîne source, utilisez ce qui suit :

```
myStr = myStr.toUpperCase();
```

Lorsque vous concaténez des chaînes, vous réunissez deux chaînes de manière séquentielle pour en former une seule. Par exemple, vous pouvez utiliser l'opérateur d'addition (+) pour concaténer deux chaînes. L'exemple suivant présente la procédure à suivre pour concaténer deux chaînes.

Pour concaténer deux chaînes :

1. Créez un document Flash, puis enregistrez-le sous le nom de **concat.fla**.
2. Ajoutez le code suivant à l'image 1 du scénario :

```
var str1:String = "green";
var str2:String = str1 + "ish";
trace(str2); // greenish
//
var str3:String = "yellow";
str3 += "ish";
trace(str3); // yellowish
```

   Le code précédent présente deux méthodes de concaténation de chaînes. La première méthode utilise l'opérateur d'addition (+) pour joindre la chaîne `str1` et la chaîne `"ish"`. La deuxième méthode utilise l'opérateur d'addition et d'affectation (+=) pour concaténer la chaîne `"ish"` avec la valeur courante de `str3`.

3. Sauvegardez le document Flash et sélectionnez Contrôle > Tester l'animation.

Vous pouvez aussi utiliser la méthode `concat()` de la classe String pour concaténer des chaînes. Ceci est illustré par l'exemple suivant :

Pour concaténer deux chaînes à l'aide de la méthode concat() :

1. Créez un document Flash, puis enregistrez-le sous le nom de **concat2.fla**.
2. Ajoutez le code suivant à l'image 1 du scénario :

```
var str1:String = "Bonjour";
var str2:String = "from";
var str3:String = "Paris";
var str4:String = str1.concat(" ", str2, " ", str3);
trace(str4); // Bonjour de Paris
```

3. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

Si vous utilisez l'opérateur d'addition (+) (ou l'opérateur d'addition et d'affectation [+=]) avec un objet chaîne et un autre objet, ActionScript convertit automatiquement les objets non chaînes en chaînes afin d'évaluer l'expression. Cette conversion est décrite dans l'exemple de code suivant :

```
var version:String = "Flash Player ";
var rel:Number = 9;
version = version + rel;
trace(version); // Flash Player 9
```

Vous pouvez toutefois utiliser des parenthèses pour forcer l'opérateur d'addition (+) à procéder à une évaluation arithmétique, comme illustré dans le code ActionScript suivant :

```
trace("Total: $" + 4.55 + 1.46); // Total : $4.551.46
trace("Total: $" + (4.55 + 1.46)); // Total : $6.01
```

Vous pouvez utiliser la méthode `split()` pour créer un tableau de sous-chaînes d'une chaîne, lequel est divisé en fonction d'un caractère délimiteur. Par exemple, vous pouvez segmenter une chaîne séparée par des virgules ou des tabulations en plusieurs chaînes.

Par exemple, le code suivant présente la division d'un tableau en sous-chaînes à l'aide du caractère esperluette (&) en tant que délimiteur.

Pour créer un tableau de sous-chaînes segmenté par un délimiteur :

1. Créez un document Flash, puis enregistrez-le sous le nom de **strsplit.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
var queryStr:String = "first=joe&last=cheng&title=manager&startDate=3/6/
  65";
var params:Array = queryStr.split("&", 2);
trace(params); // first=joe,last=cheng
    /* params est défini sur un tableau avec deux éléments :
       params[0] == "first=joe"
       params[1] == "last=cheng"
    */
```

3. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

CONSEIL Le deuxiéme paramètre de la méthode `split()` définit la taille maximale du tableau. Si vous ne souhaitez pas limiter la taille du tableau créé par la méthode `split()`, vous pouvez omettre le deuxième paramètre.

CONSEIL Le moyen le plus simple d'analyser une chaîne de requête (une chaîne délimitée par les caractères & et =) consiste à utiliser la méthode `LoadVars.decode()`, comme illustré dans l'ActionScript suivant :

```
var queryStr:String = "first=joe&last=cheng&title=manager&startDate=3/6/
  65";
var my_lv:LoadVars = new LoadVars();
my_lv.decode(queryStr);
trace(my_lv.first); // joe
```

Pour plus d'informations sur l'utilisation des opérateurs avec les chaînes, consultez « Utilisation d'opérateurs avec des chaînes », à la page 151.

Pour un exemple de fichier source, strings.fla, qui vous décrit comment construire un traitement de texte simple qui compare et recouvre des sélections de chaînes et de sous-chaînes, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Strings afin d'accéder à l'exemple.

Renvoi de sous-chaînes

Les méthodes `substr()` et `substring()` de la classe String sont identiques. Toutes deux renvoient une sous-chaîne d'une chaîne et utilisent deux paramètres. Dans les deux méthodes, le premier paramètre est la position du caractère initial dans la chaîne concernée. Toutefois, dans la méthode `substr()`, le deuxième paramètre est la *longueur* de la sous-chaîne à renvoyer, alors que dans la méthode `substring()`, le deuxième paramètre est la position du caractère *final* de la sous-chaîne (qui ne figure pas dans la chaîne renvoyée). Cet exemple illustre la différence entre ces deux méthodes :

Pour rechercher une sous-chaîne par la position d'un caractère :

1. Cééez un document Flash, puis enregistrez-le sous le nom de **substring.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
var myStr:String = "Hello from Paris, Texas!!!";
trace(myStr.substr(11,15)); // Paris, Texas!!!
trace(myStr.substring(11,15)); // Pari
```

   La première méthode, `substr()`, renvoie une chaîne de 15 caractères de long à partir du onzième caractère. La deuxième méthode, `substring()`, renvoie une chaîne de quatre caractères de longue en tenant compte de tous les caractères situés entre le onzième et le quinzième index.
3. Ajoutez l'ActionScript suivant sous le code ajouté à l'étape précédente :

```
trace(myStr.slice(11, 15)); // Pari
trace(myStr.slice(-3, -1)); // !!
trace(myStr.slice(-3, 26)); // !!!
trace(myStr.slice(-3, myStr.length)); // !!!
trace(myStr.slice(-8, -3)); // Texas
```

   La méthode `slice()` fonctionne de la même façon que la méthode `substring()`. Lorsque deux nombres entiers non négatifs sont donnés comme paramètres, le fonctionnement est le même. Toutefois, la méthode `slice()` peut accepter des entiers négatifs comme paramètres.
4. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

REMARQUE	Vous pouvez associer des entiers positifs et négatifs comme paramètres de la méthode `slice()`.

Vous pouvez utiliser les méthodes `indexOf()` et `lastIndexOf()` pour localiser des sous-chaînes correspondantes au sein d'une chaîne, comme illustré dans l'exemple suivant.

Pour trouver la position des caractères d'une sous-chaîne correspondante :

1. Créez un document Flash, puis enregistrez-le sous le nom de **indexof.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
var myStr:String = "The moon, the stars, the sea, the land";
trace(myStr.indexOf("the")); // 10
trace(myStr.indexOf("the", 11)); // 21
```

   Le premier index du mot *the* débute au 10ème caractère car la méthode `indexOf()` est sensible à la casse ; par conséquent, la première occurrence de *The* n'est pas prise en compte. Vous pouvez aussi spécifier un second paramètre de la méthode `indexOf()` pour indiquer la position d'index dans la chaîne à partir de laquelle la recherche doit débuter. Dans le code précédent, Flash recherche le premier index du mot *the* qui apparaît après le 11ème caractère.

3. Ajoutez l'ActionScript suivant sous le code que vous avez ajouté à l'étape précédente :

```
trace(myStr.lastIndexOf("the")); // 30
trace(myStr.lastIndexOf("the", 29)); // 21
```

   La méthode `lastIndexOf()` trouve la dernière occurrence d'une sous-chaîne dans la chaîne. Par exemple, au lieu de rechercher un caractère ou une sous-chaîne à partir du début d'une chaîne, `lastIndexOf()` part de la fin de la chaîne et revient en arrière. Comme avec la méthode `indexOf()`, si vous incluez un deuxième paramètre à l'aide de la méthode `lastIndexOf()`, la recherche est effectuée à partir de cette position d'index, même si avec `lastIndexOf()`, la chaîne est analysée à rebours (de droite à gauche).

4. Choisissez Contrôle > Tester l'animation pour tester votre document Flash.

CONSEIL	Les méthodes `indexOf()` et `lastIndexOf()` sont sensibles à la casse.

Pour un exemple de fichier source, strings.fla, qui vous décrit comment construire un traitement de texte simple qui compare et recouvre des sélections de chaînes et de sous-chaînes, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Strings afin d'accéder à l'exemple.

CHAPITRE 12

Animation, Filtres et Dessins 12

Ce chapitre explique comment ajouter des animations à vos applications Flash CS3 Professional avec ActionScript au lieu (ou en plus) des animations basées sur des scénarios avec interpolations de mouvements ou de formes. L'utilisation d'un code pour créer des animations et des effets réduit souvent la taille des fichiers de votre application, une fois celle-ci terminée, et peut également améliorer les performances et la cohérence de vos animations. Parfois, les animations basées sur ActionScript peuvent même réduire votre charge de travail : le code peut être plus rapide à écrire et il peut aisément être appliqué simultanément à plusieurs occurrences ou réutilisé dans d'autres applications. Ce chapitre vous explique aussi comment créer des animations en utilisant les notions de base d'ActionScript, les classes Tween et TransitionManager, l'API de dessin, des classes de filtre et des modes de fondu.

Vous pouvez utiliser l'API de dessin, c'est-à-dire les méthodes de dessin de la classe MovieClip, pour ajouter des animations et des dessins. Ces méthodes mettent à votre disposition un code pour créer des lignes, des remplissages et des formes, sans passer par les outils de dessin de l'outil de programmation.

Les filtres et autres effets visuels sont aussi importants dans de nombreuses applications Flash pour appliquer rapidement un effet et l'animer. Vous pouvez employer un code pour ajouter et animer des effets de filtre, des modes de fondu et des images bitmap.

Ce chapitre comporte les sections suivantes, qui vous expliquent comment utiliser ActionScript pour créer des animations et ajouter des effets et comment utiliser l'API de dessin pour dessiner dans ActionScript :

Mise en script d'animation avec ActionScript 2.0

Vous pouvez utiliser ActionScript pour ajouter des animations à vos applications Flash, plutôt que de créer des interpolations de mouvements ou de formes sur un scénario. Les sections suivantes vous expliquent comment utiliser le code pour animer des occurrences, par exemple en modifiant la transparence et l'apparence d'une occurrence, et déplacer l'occurrence à travers la scène.

Pour plus d'informations sur l'utilisation des classes Tween et TransitionManager en vue d'automatiser des animations à base de code, consultez la *Référence du langage des composants ActionScript 2.0*. Ces classes vous permettent d'ajouter des équations d'accélération avancées et des animations de transition à des occurrences de clip dans votre application.
Sans ces classes prédéfinies, il peut être difficile de recréer ces effets avec ActionScript, car le code que vous devez utiliser suppose de rédiger des équations mathématiques complexes pour atteindre l'effet recherché.

Pour plus d'informations sur l'animation des dessins créés à l'aide de code, consultez la section « Dessin à l'aide du code ActionScript », à la page 536.

Les sections suivantes expliquent comment scripter des animations :

- « A propos des animations et des cadences d'images », à la page 465
- « Application d'effets de fondu à des objets à l'aide de code », à la page 466
- « Application d'effets de couleur et de luminosité à l'aide du code », à la page 468
- « Déplacement d'objets avec du code », à la page 471
- « Déplacement panoramique d'une image à l'aide du code », à la page 472

Pour un exemple de fichier source animation.fla qui décrit la manipulation des tableaux à l'aide du code ActionScript, consultez la page d'exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Animation afin d'accéder à l'exemple.

Pour des exemples d'applications de galerie de photos, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Galleries afin d'accéder à ces exemples.

- gallery_tree.fla
- gallery_tween.fla

Ces fichiers vous montrent comment utiliser ActionScript pour contrôler des clips dynamiquement tout en chargeant des fichiers image dans un fichier SWF, avec des animations scriptées.

A propos des animations et des cadences d'images

Avant d'ajouter une animation à une application, il vous faut tenir compte de la cadence d'images à imposer à votre fichier FLA. Vous devez absolument penser à la cadence d'images lorsque vous travaillez sur des animations, car elle peut avoir une incidence sur les performances de votre fichier SWF et de l'ordinateur qui les lit. Une cadence d'images trop élevée risque de perturber le processeur, en particulier si vous employez des actifs multiples ou si vous utilisez ActionScript pour créer une animation.

Il faut également prêter attention à la définition de la cadence d'images, car celle-ci peut avoir une incidence sur la fluidité de la lecture de votre animation. Par exemple, une animation définie à 12 images par seconde (ips) dans l'inspecteur des propriétés va lire 12 images à la seconde. Si la cadence du document est définie à 24 ips, l'animation semble plus fluide que si elle est lue à 12 ips. Toutefois, une animation à 24 ips s'exécute bien plus rapidement qu'une animation à 12 ips, de sorte que sa durée totale (en secondes) est plus courte. En conséquence, si vous devez créer une animation de 5 secondes à une cadence supérieure, il vous faut ajouter des images supplémentaires pour remplir cette durée de 5 secondes (ce qui accroît la taille totale du fichier de votre animation). Une animation de 5 secondes à 24 ips génère normalement un fichier de taille supérieure à celui d'une animation de 5 secondes à 12 ips.

REMARQUE

Si vous utilisez un gestionnaire d'événements `onEnterFrame` pour créer des animations scriptées, l'animation s'exécute à la cadence du document, tout comme si vous aviez créé une interpolation de mouvement sur un scénario. Au lieu du gestionnaire d'événements `onEnterFrame`, il est possible d'utiliser `setInterval` (consultez la section relative à la fonction setInterval dans le Guide de référence du langage ActionScript 2.0). En ce cas, vous ne dépendez plus des cadences d'images, mais appelez des fonctions à intervalles spécifiés. Comme avec `onEnterFrame`, plus vous utilisez `setInterval` pour appeler une fonction, plus votre animation consomme de ressources sur votre processeur.

Utilisez la cadence d'images la plus basse possible, autorisant une lecture fluide de votre animation à l'exécution, de manière à réduire la charge imposée au processeur de l'utilisateur. Evitez de définir une cadence supérieure à 30-40 ips : les cadences élevées imposent une forte charge aux processeurs et ne modifient guère, voire pas du tout l'apparence des animations à l'exécution.

De même, si vous travaillez sur une animation avec scénario, veillez à définir une cadence d'images aussitôt que possible dans le processus de développement. Lorsque vous testez le fichier SWF, vérifiez la durée de votre animation, ainsi que la taille du fichier SWF. La cadence d'images a une forte incidence sur la vitesse de l'animation.

Application d'effets de fondu à des objets à l'aide de code

Lorsque vous travaillez avec des clips sur la scène, vous pouvez choisir de leur appliquer des effets de fondu en entrée ou en sortie, sans avoir à activer ou désactiver leur propriété `_visible`. L'exemple suivant explique comment utiliser un gestionnaire d'événements `onEnterFrame` pour animer un clip.

Application d'effets de fondu à un clip à l'aide de code

1. Créez un document Flash appelé **fade1.fla**.
2. Exécutez un dessin sur la scène à l'aide des outils de dessin ou importez une image vers la scène (fichier > Importer > Importer dans la scène).
3. Sélectionnez le contenu sur la scène, puis choisissez Modification > Convertir en symbole.
4. Sélectionnez l'option Clip et cliquez sur OK pour créer le symbole.
5. Sélectionnez l'occurrence de clip sur la scène et tapez **img1_mc** dans le champ Nom de l'occurrence de l'inspecteur des propriétés.
6. Sélectionnez l'image 1 du scénario, puis ajoutez le code suivant dans le panneau Actions :

```
img1_mc.onEnterFrame = function() {
  img1_mc._alpha -= 5;
  if (img1_mc._alpha <= 0) {
    img1_mc._visible = false;
    delete img1_mc.onEnterFrame;
  }
};
```

Ce code utilise un gestionnaire d'événements `onEnterFrame`, invoqué de façon répétée à la cadence du fichier SWF. Le nombre d'appels au gestionnaire d'événements par seconde dépend de la cadence d'images définie pour le document Flash. Si la cadence est de 12 images par seconde (ips), le gestionnaire d'événements `onEnterFrame` est invoqué 12 fois par seconde. Pareillement, si la cadence d'images du document Flash est de 30 ips, le gestionnaire d'événements est invoqué 30 fois par seconde.

7. Choisissez Contrôle > Tester l'animation pour tester le document.

   Le clip que vous avez ajouté sur la scène disparaît lentement en fondu.

Vous pouvez également modifier la propriété `_alpha` via la fonction `setInterval()` sans passer par un gestionnaire d'événements `onEnterFrame`, comme dans l'exemple suivant.

Application d'effets de fondu à un objet à l'aide de la fonction setInterval()

1. Créez un document Flash appelé **fade2.fla**.
2. Exécutez un dessin sur la scène à l'aide des outils de dessin ou importez une image vers la scène (fichier > Importer > Importer dans la scène).
3. Sélectionnez le contenu sur la scène, puis choisissez Modification > Convertir en symbole.
4. Sélectionnez l'option Clip et cliquez sur OK pour créer le symbole.
5. Sélectionnez l'occurrence de clip sur la scène et tapez **img1_mc** dans le champ Nom de l'occurrence de l'inspecteur des propriétés.
6. Sélectionnez l'image 1 du scénario, puis ajoutez le code suivant dans le panneau Actions :

```
var alpha_interval:Number = setInterval(fadeImage, 50, img1_mc);
function fadeImage(target_mc:MovieClip):Void {
  target_mc._alpha -= 5;
  if (target_mc._alpha <= 0) {
    target_mc._visible = false;
    clearInterval(alpha_interval);
  }
}
```

   Le comportement de la fonction `setInterval()` diffère légèrement de celui du gestionnaire d'événements `onEnterFrame`, puisque `setInterval()` indique précisément à Flash à quelle fréquence le code doit appeler une fonction particulière. Dans cet exemple de code, la fonction définie par l'utilisateur `fadeImage()` est appelée toutes les 50 millisecondes (20 fois par seconde). La fonction `fadeImage()` décrémente la valeur de la propriété `_alpha` courante du clip. Lorsque la valeur `_alpha` est égale ou inférieure à zéro, l'intervalle est effacé, ce qui a pour effet de stopper la fonction `fadeImage()`.

7. Choisissez Contrôle > Tester l'animation pour tester le document.

   Le clip que vous avez ajouté sur la scène disparaît lentement en fondu.

Pour plus d'informations sur les fonctions définies par l'utilisateur, consultez la section « Définition des fonctions globales et de scénario », à la page 182. Pour plus d'informations sur le gestionnaire d'événements `onEnterFrame`, consultez la section `onEnterFrame (gestionnaire MovieClip.onEnterFrame)` dans le *Guide de référence du langage ActionScript 2.0*. Pour plus d'informations sur la fonction `setInterval()`, consultez la section `fonction setInterval` dans le *Guide de référence du langage ActionScript 2.0*.

Pour un exemple de fichier source animation.fla dans Flash, consultez la page d'exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Animation afin d'accéder à l'exemple.

Application d'effets de couleur et de luminosité à l'aide du code

En plus d'utiliser ActionScript pour définir et animer des fondus alpha (voir « Application d'effets de fondu à des objets à l'aide de code », à la page 466), vous pouvez animer différents effets de couleur et de luminosité à l'aide de code en utilisant le panneau Filtres de l'inspecteur des propriétés.

L'exemple ci-dessous explique comment charger une image JPEG et appliquer un filtre de transformation de couleurs, qui modifie les canaux rouges et verts à mesure que le pointeur de la souris se déplace le long de l'axe *x* et de l'axe *y*.

Modification des canaux de couleur d'un objet avec ActionScript :

1. Créez un document Flash appelé **colorTrans.fla**.
2. Sélectionnez l'image 1 du scénario, puis ajoutez le code suivant dans le panneau Actions :

```
import flash.geom.Transform;
import flash.geom.ColorTransform;

var imageClip:MovieClip = this.createEmptyMovieClip("imageClip", 1);
var clipLoader:MovieClipLoader = new MovieClipLoader();
clipLoader.loadClip("http://www.helpexamples.com/flash/images/
  image1.jpg", imageClip);

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  var transformer:Transform = new Transform(imageClip);
  var colorTransformer:ColorTransform = transformer.colorTransform;
  colorTransformer.redMultiplier = (_xmouse / Stage.width) * 1;
  colorTransformer.greenMultiplier = (_ymouse / Stage.height) * 1;
  transformer.colorTransform = colorTransformer;
}
Mouse.addListener(mouseListener);
```

3. Choisissez Contrôle > Tester l'animation pour tester le document, puis déplacez le pointeur de la souris à travers la scène.

   Le fichier image qui est chargé transforme les couleurs à mesure que vous déplacez le pointeur de la souris.

Vous pouvez également utiliser la classe ColorMatrixFilter pour convertir une image couleur en image noir et blanc, comme dans l'exemple suivant.

Utilisation de la classe ColorMatrixFilter pour modifier une image en une image en nuances de gris

1. Créez un document Flash appelé **grayscale.fla**.
2. Sélectionnez l'image 1 du scénario, puis ajoutez le code suivant dans le panneau Actions :

```
import flash.filters.ColorMatrixFilter;
System.security.allowDomain("http://www.helpexamples.com");
var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function(target_mc:MovieClip):Void {
  var myElements_array:Array = [0.3, 0.59, 0.11, 0, 0,
       0.3, 0.59, 0.11, 0, 0,
       0.3, 0.59, 0.11, 0, 0,
       0, 0, 0, 1, 0];
  var myColorMatrix_filter:ColorMatrixFilter = new
  ColorMatrixFilter(myElements_array);
  target_mc.filters = [myColorMatrix_filter];
}
this.createEmptyMovieClip("img_mc", this.getNextHighestDepth());
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcl_obj);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);
```

   Le code précédent commence par importer la classe ColorMatrixFilter, puis crée un objet écouteur qui sera utilisé avec une nouvelle occurrence MovieClipLoader créée dans un prochain code. Ensuite, une occurrence de clip est créée, sous le nom d'occurrence `img_mc`, ainsi qu'une occurrence de chargeur de clip, sous le nom d'occurrence `img_mcl`. Enfin, le clip source est chargé dans le clip `img_mc` sur la scène. Une fois l'image complètement chargée, le gestionnaire d'événements onLoadInit est appelé et joint un filtre matrice de couleurs à l'image chargée.

3. Choisissez Contrôle > Tester l'animation pour tester le document.

   L'image que vous chargez sur la scène se transforme en une image en nuances de gris. Visualisez l'image en ligne (`http://www.helpexamples.com/flash/images/image1.jpg`) afin de voir la couleur originale de l'image.

Vous pouvez également définir la luminosité d'une image à l'aide du code ActionScript, comme indiqué dans l'exemple suivant.

Modification de la luminosité d'une image

1. Créez un document Flash appelé **brightness.fla**.
2. Sélectionnez l'image 1 du scénario, puis ajoutez le code suivant dans le panneau Actions :

```
import flash.filters.ColorMatrixFilter;
System.security.allowDomain("http://www.helpexamples.com/");
var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function(target_mc:MovieClip):Void {
  var myElements_array:Array = [1, 0, 0, 0, 100,
       0, 1, 0, 0, 100,
       0, 0, 1, 0, 100,
       0, 0, 0, 1, 0];
  var myColorMatrix_filter:ColorMatrixFilter = new
  ColorMatrixFilter(myElements_array);
  target_mc.filters = [myColorMatrix_filter];
}
this.createEmptyMovieClip("img_mc", this.getNextHighestDepth());
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcl_obj);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image2.jpg",
  img_mc);
```

   Ce bloc de code utilise la classe MovieClipLoader pour charger un JPEG externe. Une fois l'image complètement chargée, le gestionnaire d'événements onLoadInit de la classe MovieClipLoader est appelé et modifie la luminosité de l'image sur 100 à l'aide du filtre ColorMatrixFilter.

3. Choisissez Contrôle > Tester l'animation pour tester le document.

   La luminosité de l'image que vous chargez dans le fichier SWF change lorsque vous testez le fichier SWF. Visualisez l'image en ligne (`http://www.helpexamples.com/flash/images/image2.jpg`) afin de voir l'apparence originale de l'image.

Pour un exemple source d'animation scriptée dans Flash, animation.fla, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Animation afin d'accéder à l'exemple.

Pour des exemples d'applications de galerie de photos, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Galleries afin d'accéder à ces exemples.

- gallery_tree.fla
- gallery_tween.fla

Ces fichiers vous montrent comment utiliser ActionScript pour contrôler des clips dynamiquement tout en chargeant des fichiers image dans un fichier SWF, avec des animations scriptées.

Déplacement d'objets avec du code

Le déplacement d'un objet à l'aide d'ActionScript est analogue à la modification de la propriété `_alpha` d'un objet : il vous suffit de modifier la propriété `_x` ou `_y`.

L'exemple suivant anime une image JPEG chargée dynamiquement et la fait glisser horizontalement à travers la scène.

Déplacement d'une occurrence sur la scène à l'aide de code

1. Créez un document Flash appelé **moveClip.fla**.
2. Modifiez la cadence d'images du document à 24 ips dans l'inspecteur des propriétés.

   L'animation est bien plus fluide si vous utilisez une cadence élevée, par exemple 24 ips.
3. Sélectionnez l'image 1 du scénario, puis ajoutez le code suivant dans le panneau Actions :

```
// créer une occurrence de clip
this.createEmptyMovieClip("img1_mc", 10);
var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function (target_mc:MovieClip):Void {
  target_mc._x = Stage.width;
  target_mc.onEnterFrame = function() {
    target_mc._x -= 3; // décrémenter la position _x courante
                       // de 3 pixels
    if (target_mc._x <= 0) {
      target_mc._x = 0;
      delete target_mc.onEnterFrame;
    }
  };
};
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcl_obj);
// charger une image dans le clip
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img1_mc);
```

   Cet exemple de code charge une image externe à partir d'un serveur Web distant et, une fois l'image complètement chargée, l'anime horizontalement à travers la scène. Au lieu d'utiliser un gestionnaire d'événements `onEnterFrame`, vous pouvez utiliser la fonction `setInterval()` pour animer l'image.
4. Choisissez Contrôle > Tester l'animation pour tester le document.

   L'image se charge et s'anime sur la scène, depuis la droite vers le coin supérieur gauche.

Pour plus d'informations sur l'utilisation d'un gestionnaire d'événements `onEnterFrame` ou de la fonction `setInterval()`afin d'animer une image, consultez la section « Application d'effets de fondu à des objets à l'aide de code », à la page 466.

Pour un exemple de fichier source d'animation scriptée dans Flash, animation.fla, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Animation afin d'accéder à l'exemple.

Pour des exemples d'applications de galerie de photos, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Galleries afin d'accéder à ces exemples.

- gallery_tree.fla
- gallery_tween.fla

Ces fichiers vous montrent comment utiliser le code ActionScript pour contrôler des clips dynamiquement tout en chargeant des fichiers image dans un fichier SWF, avec des animations scriptées.

Déplacement panoramique d'une image à l'aide du code

Grâce à ActionScript, vous pouvez facilement afficher de grands panoramiques d'images dans vos documents Flash. Cette fonction est particulièrement utile lorsque votre image ne rentre pas parfaitement dans la scène ou lorsque vous cherchez à créer un effet d'animation en déplaçant le plan panoramique d'un clip d'un côté à l'autre de la scène. Par exemple, lorsqu'une image panoramique est plus grande que votre scène mais que vous ne souhaitez pas la réduire, ni agrandir les dimensions de la scène, vous pouvez créer un clip qui servira de masque à l'image.

L'exemple suivant explique comment masquer un clip dynamiquement et utiliser un gestionnaire d'événements `onEnterFrame` pour animer une image derrière le masque.

Déplacement panoramique d'une occurrence sur la scène à l'aide de code :

1. Créez un document Flash appelé **panel.fla**.
2. Modifiez la cadence d'images du document à 24 ips dans l'inspecteur des propriétés.

   L'animation est bien plus fluide si vous utilisez une cadence élevée, par exemple 24 ips.
3. Sélectionnez l'image 1 du scénario, puis ajoutez le code suivant dans le panneau Actions :

```
System.security.allowDomain("http://www.helpexamples.com/");
// initialiser des variables
var direction:Number = -1;
var speed:Number = 5;
// créer un clip pour y charger une image
this.createEmptyMovieClip("img_mc", 10);
```

```
// créer un clip à utiliser comme masque
this.createEmptyMovieClip("mask_mc", 20);
// utiliser l'API de dessin pour dessiner/créer un masque
with (mask_mc) {
  beginFill(0xFF0000, 0);
  moveTo(0, 0);
  lineTo(300, 0);
  lineTo(300, 100);
  lineTo(0, 100);
  lineTo(0, 0);
  endFill();
}

var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function(target_mc:MovieClip) {
  // définir le masque du clip cible sur mask_mc
  target_mc.setMask(mask_mc);
  target_mc.onEnterFrame = function() {
    target_mc._x += speed * direction;
    // si target_mc se retrouve sur un bord, inverser le sens
    // de l'animation
    if ((target_mc._x <= -(target_mc._width-mask_mc._width)) ||
  (target_mc._x >= 0)) {
      direction *= -1;
    }
  };
};

var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.addListener(mcl_obj);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);
```

Dans cet exemple, la première section du code définit deux variables : `direction` et `speed`. La variable `direction` contrôle le défilement de l'image masquée de gauche à droite (`1`) ou de droite à gauche (`-1`). La variable `vitesse` contrôle le nombre de pixels déplacés à chaque appel du gestionnaire d'événements `onEnterFrame`. Si le nombre est très grand, l'animation se déplace plus rapidement et, dans ce cas, risque d'apparaître moins fluide.

La section de code suivante crée deux clips vides : `img_mc` et `mask_mc`. Un rectangle de 300 x 100 pixels est tracé dans le clip `mark_mc` à l'aide de l'API de dessin. Un nouvel objet (`mcl_obj`) est ensuite créé pour servir d'écouteur de l'occurrence MovieClipLoader créée dans le dernier bloc de code. Cet objet définit un écouteur pour l'événement `onLoadInit`, masque l'image chargée dynamiquement et fixe le défilement de l'animation. Une fois que l'image atteint le bord gauche ou droit du masque, l'animation est inversée.

Le dernier bloc de code définit une occurrence de MovieClipLoader, spécifie l'objet écouteur précédemment créé et commence à charger l'image JPEG dans le clip `img_mc`.

4. Choisissez Contrôle > Tester l'animation pour tester le document.

   L'image se charge, puis s'anime d'avant en arrière dans un mouvement panoramique (latéral). L'image est masquée à l'exécution. Pour voir l'image originale, affichez-la en ligne (`http://www.helpexamples.com/flash/images/image1.jpg`).

A propos de la mise en cache bitmap, du défilement et des performances

Flash intègre la mise en cache bitmap, qui vous permet d'améliorer les performances des clips immuables dans vos applications. Si vous définissez la propriété `MovieClip.cacheAsBitmap` ou `Button.cacheAsBitmap` sur `true`, Flash Player place en mémoire cache une version bitmap interne de l'occurrence de clip ou de bouton. Cette propriété peut améliorer les performances des clips incluant un contenu vectoriel complexe. Toutes les données vectorielles d'un clip contenant un bitmap en mémoire cache sont tracées sur le bitmap, et non pas sur la scène principale.

REMARQUE	Ce bitmap est ensuite copié sur la scène principale sous forme de pixels, sans étirement ni rotation, puis accroché aux limites de pixels les plus proches. Les correspondances des pixels avec l'objet parent se font selon un rapport de 1 à 1. Si les limites du bitmap changent, le bitmap est recréé au lieu d'être étiré.

Pour plus d'informations sur la mise en cache d'occurrences de bouton ou de clip, consultez les sections suivantes dans le Chapitre 10, « Utilisation des clips » :

- « Mise en cache et parcours de clips à l'aide d'ActionScript », à la page 353
- « Mise en cache d'un clip », à la page 357
- « Réglage de l'arrière-plan d'un clip », à la page 360

La propriété `cacheAsBitmap` s'utilise de préférence avec des clips dont le contenu est principalement statique et qui sont rarement redimensionnés ou pivotés. Avec ce type de clips, la propriété `cacheAsBitmap` peut améliorer les performances lors de la conversion du clip (lorsque les positions *x* et *y* sont modifiées). Pour plus d'informations sur l'utilisation de cette fonction, consultez « Quand activer la mise en cache », à la page 355.

Pour des exemples sur l'application de mises en cache bitmap dans une occurrence et dans du texte de défilement, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Les exemples suivants sont disponibles :

- cacheBitmap.fla ; Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/CacheBitmap.
- aliasing.fla ; Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Advanced Anti-Aliasing.

A propos des classes Tween et TransitionManager

Lorsque vous installez Flash CS3 Professional, vous installez également les classes Tween et TransitionManager. Cette section vous explique comment utiliser ces deux classes puissantes avec des clips et des composants Adobe pour ajouter en toute simplicité une animation à vos fichiers SWF.

Si vous créez un diaporama ou une application de formulaires avec Flash (ActionScript 2.0 uniquement), vous pouvez sélectionner des comportements qui ajoutent différents types de transitions entre les diapositives, de la même façon que dans une présentation PowerPoint. Vous ajoutez cette fonctionnalité dans une application d'écrans à l'aide des classes Tween et TransitionManager, qui génèrent un code ActionScript animant les écrans selon le comportement choisi.

Vous pouvez également utiliser les classes Tween et TransitionManager en dehors d'un document à base d'écrans dans Flash. Vous pouvez par exemple utiliser ces classes avec le jeu de composants de la version 2 de l'architecture des composants Adobe ou avec des clips. Si vous souhaitez modifier l'animation d'un composant ComboBox, utilisez la classe TransitionManager pour ajouter une certaine *accélération* à l'ouverture du menu. Ce terme fait ici référence à une accélération ou une décélération progressive survenant pendant l'animation et destinée à améliorer son réalisme. Vous pouvez également utiliser les classes Tween et TransitionManager pour créer votre propre système de menus animé, sans passer par l'interpolation de mouvement ou la rédaction d'un code personnalisé.

REMARQUE	Les classes Tween et TransitionManager ne sont accessibles que sous ActionScript 2.0, mais sont disponibles dans Flash.

Pour plus d'informations sur les méthodes et propriétés de la classe Tween, consultez la *Référence du langage des composants ActionScript 2.0*. Pour plus d'informations sur les méthodes et propriétés de la classe TransitionManager, consultez la *Référence du langage des composants ActionScript 2.0*. Pour plus d'informations sur l'utilisation des paquets, consultez « Fonctionnement des paquets de filtres », à la page 493.

Pour un exemple de fichier source tweenProgress.fla qui utilise les classes Tween et TransitionManager, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Tween ProgressBar afin d'accéder à l'exemple.

Pour plus d'informations sur les classes Tween et TransitionManager, consultez les sections suivantes :

Ajout d'interpolations et de transitions à un fichier dans Flash Professional 8 (Flash Professional 8 uniquement)

REMARQUE

Cette section explique comment ajouter des interpolations et des transitions à un diaporama Flash Professional et décrit la manière dont elles apparaissent aux utilisateurs de Flash Professional. Vous pouvez toutefois ajouter des transitions et des interpolations à vos applications Flash Basic 8 (ou Flash Professional 8) avec du code. Les sections suivantes vous présentent des exemples à cet effet.

Les classes Tween et TransitionManager sont conçues pour vous aider à ajouter des animations à des parties de votre fichier SWF à l'aide du code ActionScript simple. L'environnement de programmation de Flash contient des comportements qui vous permettent d'utiliser ces classes prédéfinies pour les transitions d'une application composée d'écrans. Pour créer un diaporama ou une application de formulaires, vous pouvez sélectionner des comportements qui ajoutent différents types de transitions entre les diapositives.

Avant de commencer à utiliser ces transitions avec des clips dans Flash, découvrez leurs capacités dans une application à base d'écrans.

Pour voir le code ActionScript qui crée une transition dans un diaporama :

1. Choisissez Fichier > Nouveau pour créer un diaporama dans Flash Professional 8.
2. Sélectionnez Diaporama Flash dans l'onglet Général, puis cliquez sur OK.
3. Choisissez Fenêtre > Comportements pour ouvrir le panneau Comportements.
4. Cliquez sur Ajouter (+).
5. Choisissez Ecran > Transition dans le menu contextuel pour ouvrir la boîte de dialogue Transitions.
6. Sélectionnez la transition Zoom.

7. Tapez **1** dans le champ Durée.
8. Sélectionnez Rebond dans le menu contextuel Accélération.
9. Cliquez sur OK pour appliquer les paramètres et fermer la boîte de dialogue.

   L'opération ajoute environ 15 lignes de code ActionScript directement sur la diapositive. L'exemple de code suivant permet d'examiner la transition :

```
mx.transitions.TransitionManager.start(eventObj.target,
  {type:mx.transitions.Zoom, direction:0, duration:1,
  easing:mx.transitions.easing.Bounce.easeOut, param1:empty,
  param2:empty});
```

   Ce code appelle la classe TransitionManager, puis applique la transition Zoom avec la méthode d'accélération `mx.transitions.easing.Bounce.easeOut` spécifiée. Dans ce cas, la transition s'applique à la diapositive sélectionnée. Pour appliquer cet effet à un clip, vous pouvez modifier le code ActionScript à utiliser dans vos animations Flash.
   La modification du code pour travailler avec un symbole de clip est très simple : changez le premier paramètre `eventObj.target` et définissez-le sur le nom de l'occurrence de clip souhaitée.

Flash est livré avec dix transitions que vous pouvez personnaliser à l'aide des méthodes d'accélération et de plusieurs paramètres facultatifs. Cependant, n'oubliez pas que l'accélération fait ici référence à une accélération ou une décélération progressive survenant pendant l'animation et destinée à améliorer son réalisme. Par exemple, la vitesse d'une balle peut croître progressivement au début d'une animation, puis ralentir avant de s'arrêter complètement à la fin de l'animation. Les nombreuses équations qui provoquent ces accélération et décélération modifient l'animation en conséquence.

Le tableau ci-après recense les transitions incluses dans Flash Basic 8 (avec code) et dans Flash Professional 8 (avec code ou comportements) :

Transition	Description
Iris	Fait apparaître l'écran ou le clip à travers le masque animé d'une forme qui s'agrandit.
Effacement	Fait apparaître l'écran ou le clip à travers le masque animé d'une forme qui se déplace horizontalement.
Dissolution des pixels	Masque l'écran ou le clip avec des rectangles qui apparaissent ou disparaissent.
Stores	Fait apparaître l'écran ou le clip suivant à travers des rectangles qui apparaissent ou disparaissent.
Fondu	Efface ou fait apparaître l'écran ou le clip.
Vol	Glisse sur l'écran ou le clip à partir d'une direction spécifique.

Transition	Description
Zoom	Fait un zoom avant ou arrière sur l'écran ou le clip.
Compression	Redimensionne l'écran ou le clip horizontalement ou verticalement.
Pivoter	Fait pivoter l'écran ou le clip en cours.
Photo	Fait apparaître l'écran ou le clip à la manière d'un flash photographique.

Chaque transition présente de légères possibilités de personnalisation que vous pouvez appliquer à l'animation. La boîte de dialogue Transitions permet d'afficher un aperçu de l'animation avant d'appliquer l'effet à la diapositive ou au formulaire.

CONSEIL

Pour avoir un aperçu de la manière dont chaque transition s'exécute avec les différentes méthodes des classes d'accélération, double-cliquez sur Transition.swf dans le dossier *lecteur de démarrage*\Program Files\Adobe\Adobe Flash CS3*langue*\First Run\Behaviors\ ou *Disque dur Macintosh*:Applications:Adobe Flash CS3:First Run:Behaviors: de façon à ouvrir le fichier SWF dans le lecteur autonome.

Animation avec les classes TransitionManager et Tween

Vous pouvez utiliser les classes TransitionManager et Tween dans Flash pour ajouter des animations à des clips, des composants et des images à l'aide du code ActionScript. Si vous n'utilisez pas la classe TransitionManager ou Tween, vous devez écrire du code personnalisé pour animer vos clips ou modifier leur niveau de transparence (alpha) et leurs coordonnées (emplacement). Si vous souhaitez ajouter une accélération à l'animation, le code ActionScript et les équations mathématiques nécessaires deviennent rapidement complexes. Toutefois, si vous souhaitez modifier l'accélération d'une animation particulière à l'aide de ces classes prédéfinies, vous pouvez changer de classe au lieu de définir les équations nécessaires à la création d'une animation fluide.

L'exemple suivant anime un clip en le faisant grossir sur la scène via la classe TransitionManager.

Pour animer un clip avec la classe TransitionManager :

1. Choisissez Fichier > Nouveau, puis sélectionnez un document Flash.
2. Cliquez sur OK pour créer le nouveau fichier FLA.
3. Enregistrez le fichier FLA sous le nom de **zoom.fla**.

4. Choisissez Fichier > Importer > Importer dans la scène, puis sélectionnez sur votre disque l'image à importer dans le fichier FLA.

   L'image est importée dans votre fichier au format bitmap. Il vous faut donc la convertir manuellement en symbole de clip.

5. Cliquez sur Ouvrir pour importer l'image.
6. Sélectionnez l'image importée sur la scène, puis choisissez Modification > Convertir en symbole.
7. Nommez le symbole **img1** et assurez-vous de définir le comportement sur Clip.

   Par défaut, le coin supérieur gauche est défini comme point d'alignement du symbole.

8. Cliquez sur OK pour convertir l'image bitmap en clip.
9. L'image étant toujours sélectionnée, ouvrez l'inspecteur des propriétés (Fenêtre > Propriétés > Propriétés) et attribuez le nom d'occurrence **img1_mc** au clip.
10. Sélectionnez l'image 1 du scénario principal et, dans le panneau Actions, ajoutez le code ActionScript suivant :

```
mx.transitions.TransitionManager.start(img1_mc,
  {type:mx.transitions.Zoom, direction:0, duration:1,
  easing:mx.transitions.easing.Bounce.easeOut, param1:empty,
  param2:empty});
```

REMARQUE

Pour plus d'informations sur l'utilisation des paquets, consultez « Fonctionnement des paquets de filtres », à la page 493.

11. Choisissez Contrôle > Tester l'animation pour tester le clip.

    L'image s'agrandit et fait un léger rebond avant de retrouver sa taille d'origine. Si l'animation va trop vite, augmentez-en la durée (dans le code précédent) en passant d'une à deux ou trois secondes (par exemple, `duration:3`).

    Vous pouvez remarquer que l'image est ancrée dans le coin supérieur gauche et s'agrandit en direction du coin inférieur droit. L'aperçu que vous présente la boîte de dialogue Transition est légèrement différent.

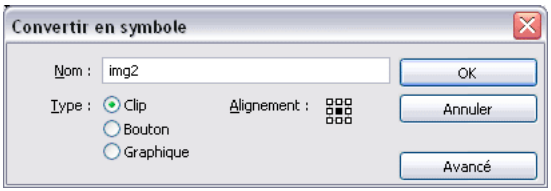
    La création d'animations complexes à l'aide des classes Tween et TransitionManager reste aisée et ne nécessite pas de création d'interpolations de mouvements ou de formes sur le scénario. Plus important encore, vous n'avez pas besoin d'élaborer des équations mathématiques complexes pour créer des méthodes d'accélération. Si vous préférez que les images s'agrandissent à partir du centre et non à partir d'un coin, vous devez modifier le point d'alignement du symbole lorsque vous convertissez l'image bitmap en symbole.

Pour agrandir des images à partir du centre :

1. Suivez la procédure indiquée à la section précédente.
2. Ouvrez le fichier zoom.fla, puis choisissez Fichier > Enregistrer sous pour enregistrer une nouvelle copie du document.

   Enregistrez le fichier sous le nom de **zoom2.fla**.
3. Faites glisser une copie du symbole bitmap du panneau Bibliothèque vers la scène, à côté du symbole de clip actuel.
4. L'image bitmap toujours sélectionnée sur la scène, appuyez sur la touche F8 pour convertir le symbole en clip.

   Nommez le symbole **img2**.
5. Dans la boîte de dialogue Convertir en symbole, cliquez sur le centre de la grille des coordonnées pour définir le point d'alignement au centre de l'image bitmap, puis cliquez sur OK.



6. Sélectionnez le nouveau clip sur la scène et entrez **img2_mc** comme nom d'occurrence dans l'inspecteur des propriétés.
7. Sélectionnez l'image 1 du scénario principal et ajoutez le code ActionScript suivant au code existant :

```
mx.transitions.TransitionManager.start(img2_mc,
  {type:mx.transitions.Zoom, direction:mx.transitions.Transition.IN,
  duration:1, easing:mx.transitions.easing.Bounce.easeOut});
```

8. Choisissez Contrôle > Tester l'animation pour tester le clip.

   Le deuxième clip s'agrandit à partir du centre du symbole, et non pas à partir du coin.

REMARQUE	Certaines transitions sont sensibles à l'emplacement du point d'alignement. La modification de ce point peut donc avoir un effet dramatique sur l'animation dans le fichier SWF. Par exemple, si le point d'alignement se trouve dans le coin supérieur gauche (emplacement par défaut) lorsque vous utilisez la transition Zoom, la transition commence à partir de cet emplacement.

Pour plus d'informations sur les méthodes et propriétés de la classe Tween et de la classe TransitionManager, consultez la *Référence du langage des composants ActionScript 2.0*.

Pour un exemple de fichier source tweenProgress.fla qui ajoute de l'animation scriptée à l'aide des classes Tween et TransitionManager, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Tween ProgressBar afin d'accéder à cet exemple.

A propos des classes et méthodes d'accélération

La section « Ajout d'interpolations et de transitions à un fichier dans Flash Professional 8 (Flash Professional 8 uniquement) », à la page 476 explique comment utiliser la classe d'accélération Bounce pour ajouter un effet de rebond au clip. Outre cette classe Bounce, Flash Basic 8 et Flash Professional 8 proposent cinq classes de ce type, recensées dans le tableau suivant :

Transition	Description
En arrière	Etend l'animation au-delà de la plage de transition à l'une ou aux deux extrémités pour créer un effet de débordement.
Rebond	Ajoute un effet de rebond à la plage de transition, à l'une ou aux deux extrémités. Le nombre de rebonds dépend de la durée, plus elle est longue, plus les rebonds sont nombreux.
Elastique	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.
Normale	Donne un effet ralenti à l'une ou aux deux extrémités. Cette fonctionnalité permet d'ajouter un effet d'accélération, de décélération, ou les deux.
Rapide	Donne un effet ralenti à l'une ou aux deux extrémités. Cet effet est similaire à l'effet Normal, mais en plus prononcé.
Aucun	Ajoute un mouvement régulier du début à la fin, sans effet, sans ralentissement, ni accélération. Cette transition est également appelée transition linéaire.

Ces six classes d'accélération présentent chacune trois méthodes d'accélération, décrites dans le tableau suivant :

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.

Pour ouvrir ces classes dans Flash ou dans votre éditeur ActionScript, ouvrez le dossier *Disque dur*\Program Files\Adobe\Adobe Flash CS3*langue*\First Run\Classes\mx\transitions\easing\ sous Windows (dans le cas d'une installation par défaut) ou le dossier *Disque dur Macintosh*:Applications:Adobe Flash CS3:First Run:Classes:mx:transitions:easing.

La procédure d'agrandissement des images décrite Ã la section « Animation avec les classes TransitionManager et Tween », à la page 478 a recours à la classe et à la méthode d'accélération mx.transitions.easing.Bounce.easeOut. Dans le dossier de votre disque dur, le fichier ActionScript fait référence à la méthode `easeOut()` dans la classe Bounce.as. Ce fichier ActionScript est stocké dans le dossier easing.

Pour plus d'informations sur les méthodes et propriétés de la classe Tween et de la classe TransitionManager, consultez la *Référence du langage des composants ActionScript 2.0*.

CONSEIL

Pour avoir un aperçu de la manière dont chaque transition s'exécute avec les différentes méthodes des classes d'accélération, double-cliquez sur Transition.swf dans le dossier *lecteur de démarrage*\Program Files\Adobe\Adobe Flash CS3*langue*\First Run\Behaviors\ ou *Disque dur Macintosh*:Applications:Adobe Flash CS3:First Run:Behaviors: de façon à ouvrir le fichier SWF dans le lecteur autonome.

Pour un exemple de fichier source tweenProgress.fla qui ajoute de l'animation scriptée à l'aide des classes Tween et TransitionManager, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Tween ProgressBar afin d'accéder à cet exemple.

A propos de la classe Tween

La classe Tween permet de déplacer, de redimensionner et d'appliquer aisément des fondus aux clips sur la scène. Le constructeur pour la classe mx.transitions.Tween possède les noms et les types de paramètres suivants :

```
function Tween(obj, prop, func, begin, finish, duration, useSeconds) {
    // code ...
}
```

`obj` est l'objet de clip ciblé par l'occurrence de Tween.

`prop` est le nom de chaîne d'une propriété dans `obj` auquel les valeurs seront interpolées.

`func` est la méthode d'accélération qui calculera un effet d'accélération pour les valeurs de propriété de l'objet interpolé.

`begin` est un chiffre indiquant la valeur de début de `prop` (la propriété de l'objet cible à interpoler).

`finish` est un chiffre indiquant la valeur de fin de `prop` (la propriété de l'objet cible à interpoler).

`duration` est un chiffre indiquant la durée du mouvement d'interpolation. Si ce paramètre n'est pas précisé, la durée sera définie par défaut sur `infinity`.

`useSeconds` est une valeur booléenne liée à la valeur définie dans le paramètre `duration`. Elle indique d'utiliser des secondes si la valeur est définie sur `true` ou des images si la valeur est définie sur `false`.

Imaginons, par exemple, que vous souhaitiez déplacer un clip à travers la scène. Vous pouvez ajouter des images-clés à un scénario et insérer une interpolation de mouvement ou de forme entre elles, écrire du code dans un gestionnaire d'événements `onEnterFrame` ou encore utiliser la fonction `setInterval()` pour appeler une fonction à intervalles réguliers. Si vous utilisez la classe Tween, une autre possibilité consiste à modifier les propriétés `_x` et `_y` d'un clip. Vous pouvez également ajouter les méthodes d'accélération décrites précédemment. Pour tirer profit de la classe Tween, vous pouvez utiliser le code ActionScript suivant :

```
new mx.transitions.Tween(ball_mc, "_x",
  mx.transitions.easing.Elastic.easeOut, 0, 300, 3, true);
```

Ce fragment de code ActionScript crée une occurrence de la classe Tween, qui anime le clip `ball_mc` sur la scène le long de l'axe *x* (de gauche à droite). Le clip s'anime de 0 à 300 pixels en trois secondes et le code ActionScript applique une méthode d'accélération élastique. La balle s'étire au-delà de 300 pixels sur l'axe *x* avant de revenir en arrière dans un mouvement fluide.

Pour un exemple de fichier source tweenProgress.fla qui ajoute de l'animation scriptée à l'aide des classes Tween et TransitionManager, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Tween ProgressBar afin d'accéder à cet exemple.

Utilisation de la classe Tween

Si vous utilisez la classe Tween en plusieurs endroits de votre document Flash, vous pouvez choisir d'utiliser une instruction `import`. Cela vous permet d'importer la classe Tween et des méthodes d'accélération au lieu de préciser les noms de classe pleinement qualifiés à chaque utilisation, comme dans l'exemple suivant.

Importation et utilisation de la classe Tween :

1. Créez un document Flash appelé **easeTween.fla**.
2. Créez un clip sur la scène.
3. Sélectionnez l'occurrence de clip sur la scène et tapez **ball_mc** dans le champ Nom de l'occurrence de l'inspecteur des propriétés.

4. Sélectionnez l'image 1 du scénario, puis ajoutez le code suivant dans le panneau Actions :

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
new Tween(ball_mc, "_x", Elastic.easeOut, Stage.width, 0, 3, true);
```

Cet exemple de code utilise deux instructions `import`. La première instruction importe uniquement la classe mx.transitions.Tween et la seconde instruction `import` utilise le raccourci générique (`*`) pour importer chacune des six classes d'accélération à l'aide d'une seule ligne de code. La seconde instruction importe un package entier de classes.

REMARQUE

Pour plus d'informations sur l'utilisation des paquets, consultez « Fonctionnement des paquets de filtres », à la page 493.

5. Choisissez Contrôle > Tester l'animation pour visualiser l'animation.

La documentation de Flash définit *package* en tant que répertoire contenant un ou plusieurs fichier(s) de classe et résidant dans un répertoire de chemin de classe désigné. Dans ce cas, le paquet réside dans le dossier C:\Program Files\Adobe\Adobe Flash CS3*langue*\First Run\Classes\mx\transitions\easing (Windows) ou dans DD:Applications:Adobe Flash CS3:First Run:Classes:mx:transitions:easing (Macintosh). Vous conviendrez sans nul doute qu'il est bien plus simple d'importer un package complet que d'importer six classes séparément. Au lieu de faire référence à la classe mx.transitions.Tween, votre code ActionScript fait directement référence à la classe Tween. De même, au lieu d'utiliser le nom de classe complet pour les classes d'accélération, mx.transitions.easing.Elastic.easeOut par exemple, il suffit de taper **Elastic.easeOut** dans votre code ActionScript. Pour plus d'informations, voir « Fonctionnement des paquets de filtres », à la page 493.

A l'aide du même code, vous définissez la propriété `_alpha` pour appliquer un fondu à des occurrences, et non plus la propriété `_x`, comme dans l'exemple de code suivant.

Application d'effets de fondu à des occurrences à l'aide de la classe Tween

1. Créez un document Flash appelé **fadeTween.fla**.
2. Créez un clip sur la scène.
3. Sélectionnez l'occurrence de clip et tapez **ball_mc** dans le champ Nom de l'occurrence de l'inspecteur des propriétés.

4. Sélectionnez l'image 1 du scénario, puis ajoutez le code suivant dans le panneau Actions :

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
new Tween(ball_mc, "_alpha", Strong.easeIn, 100, 0, 3, true);
```

   Au lieu de se déplacer sur la scène, `ball_mc` effectue maintenant un fondu et, en trois secondes, passe d'une visibilité de 100 % à une transparence complète. Pour que le symbole disparaisse plus rapidement, changez le paramètre de durée de `3` à `1` ou `2`.

5. Choisissez Contrôle > Tester l'animation pour visualiser l'animation.

   Si vous modifiez la cadence d'images du document, la lecture de l'animation devient plus fluide. Pour plus d'informations sur les animations et les cadences d'images, consultez « A propos des animations et des cadences d'images », à la page 465.

Au lieu d'utiliser des secondes, vous pouvez appliquer le fondu au symbole sur quelques images. Pour définir la durée en images plutôt qu'en secondes dans la classe Tween, modifiez le paramètre final, `useSeconds`, de `true` en `false`. En définissant ce paramètre sur `true`, vous indiquez à Flash que la durée spécifiée est exprimée en secondes. Si vous définissez le paramètre sur `false`, la durée correspond au nombre d'images à utiliser pour l'interpolation. L'exemple suivant montre comment appliquer une interpolation à des images, au lieu de secondes.

Définition d'une durée en images au lieu de secondes :

1. Créez un document appelé **framesTween.fla**.
2. Créez un clip sur la scène.
3. Sélectionnez l'occurrence de clip et tapez **ball_mc** dans le champ Nom de l'occurrence de l'inspecteur des propriétés.
4. Sélectionnez l'image 1 du scénario, puis ajoutez le code suivant dans le panneau Actions :

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
new Tween(ball_mc, "_alpha", Strong.easeIn, 100, 0, 24, false);
```

   Ce code applique un fondu à l'occurrence `ball_mc` à l'aide de la méthode d'accélération `Strong.easeIn`. Au lieu d'être appliqué pendant trois secondes, le fondu s'étale sur 24 images.

5. Choisissez Contrôle > Tester l'animation pour visualiser l'animation.

   Au bout d'un moment, le fondu s'exécute sur 24 images.

6. Revenez dans l'environnement de programmation et ouvrez l'inspecteur des propriétés
7. Modifiez la cadence du document à **24** ips.

   Si vous augmentez la cadence d'images de votre fichier FLA, le fondu s'applique plus tôt à l'occurrence. Pour plus d'informations sur les animations et les cadences d'images, consultez « A propos des animations et des cadences d'images », à la page 465.

L'utilisation d'images à la place de secondes offre plus de souplesse, mais n'oubliez pas que la durée est liée à la cadence d'images du document Flash actif. Si votre document Flash utilise une cadence de 12 images par seconde (ips), le fondu obtenu par le code précédent dure deux secondes (24 images/12 ips = 2 secondes). Par contre, si votre cadence est de 24 images par seconde, le même code donne un fondu d'une seconde (24 images/24 images/seconde = 1 seconde). Le fait de mesurer la durée en images permet de modifier significativement la vitesse d'une animation en changeant la cadence d'images du document, sans rien changer au code ActionScript.

La classe Tween offre plusieurs fonctionnalités très utiles. Par exemple, vous pouvez écrire un gestionnaire d'événements qui se déclenche une fois que l'animation est terminée, comme dans l'exemple suivant.

Déclenchement de code à la fin d'une animation :

1. Créez un document appelé **triggerTween.fla**.
2. Créez un clip sur la scène.
3. Sélectionnez l'occurrence de clip sur la scène et tapez **ball_mc** dans le champ Nom de l'occurrence de l'inspecteur des propriétés.
4. Sélectionnez l'image 1 du scénario, puis ajoutez le code suivant dans le panneau Actions :

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
var tween_handler:Tween = new Tween(ball_mc, "_alpha", Strong.easeIn,
  100, 0, 3, true);
tween_handler.onMotionFinished = function() {
  trace("onMotionFinished triggered");
};
```

   Si vous testez ce code ActionScript dans votre fichier FLA, vous obtenez le message « onMotionFinished déclenché » dans le panneau de sortie une fois que le fondu appliqué à `ball_mc` prend fin sur la scène.

5. Choisissez Contrôle > Tester l'animation pour visualiser l'animation.

   Au bout d'un moment, le fondu s'exécute sur l'occurrence. Une fois l'interpolation terminée, vous obtenez le message dans le panneau de sortie.

Pour plus d'informations sur ces fonctions, consultez le Chapitre 6, « Classes »

A propos de la poursuite d'animations à l'aide de la méthode continueTo()

« Utilisation de la classe Tween », à la page 483 explique comment utiliser la classe Tween dans vos applications. Toutefois, si vous souhaitez déplacer la balle à l'issue de l'animation initiale, vous avez au moins deux façons de procéder. La première solution consiste à animer de nouveau la balle à l'aide du gestionnaire d'événements `onMotionFinished`. Toutefois, la classe Tween offre une solution plus simple via la méthode `continueTo()`. La méthode `continueTo()` demande à l'animation interpolée de continuer à partir de sa valeur actuelle vers une nouvelle valeur, comme dans l'exemple ActionScript suivant :

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
var ball_tween:Tween = new Tween(ball_mc, "_x", Regular.easeIn, 0, 300, 3,
  true);
ball_tween.onMotionFinished = function() {
    ball_tween.continueTo(0, 3);
};
```

A la fin de l'interpolation initiale, le clip `ball_mc` reprend sa place d'origine à 0 pixel. Le code suivant présente le prototype de fonction de la méthode `continueTo()` (modifié pour plus de concision) :

```
function continueTo(finish:Number, duration:Number):Void {
    /* ignoré pour économiser de l'espace. */
}
```

Deux arguments seulement sont transmis à la méthode `continueTo()` au lieu des sept réclamés par la méthode constructeur Tween, comme dans le code suivant :

```
function Tween (obj, prop, func, begin, finish, duration, useSeconds) {
    /* ignoré pour économiser de l'espace. */
}
```

Cinq arguments sont inutiles avec la méthode `continueTo()` (`obj`, `prop`, `func`, `begin` et `useSeconds`), qui utilise les arguments que vous avez définis précédemment dans l'appel à la classe Tween. Lorsque vous appelez la méthode `continueTo()`, vous supposez que les arguments `obj`, `prop`, `func` (type d'accélération) et `useSeconds` sont les mêmes que pour l'appel précédent à la classe Tween. La méthode `continueTo()` utilise la valeur `finish` de l'appel à la classe Tween au lieu de spécifier une valeur pour l'argument `begin`, comme dans le code ActionScript suivant :

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
var ball_tween:Tween = new Tween(ball_mc, "_x", Regular.easeIn, 0, 300, 3,
  true);
ball_tween.onMotionFinished = function() {
     ball_tween.continueTo(0, 3);
};
```

Ce code déplace l'occurrence `ball_mc` le long de l'axe *x*-de 0 pixel à 300 pixels en trois secondes. A la fin de l'animation, le gestionnaire d'événements `onMotionFinished` se déclenche et appelle la méthode `continueTo()`. La méthode `continueTo()` indique à l'objet cible (`ball_mc`) de continuer l'animation à partir de sa position courante et pendant trois secondes le long de l'axe *x* à 0 pixel et d'utiliser la même méthode d'accélération. Vous utilisez les valeurs spécifiées dans l'appel à la méthode constructeur Tween pour tous les paramètres que vous ne définissez pas dans la méthode `continueTo()`. Si vous n'indiquez pas de durée, la méthode `continueTo()` utilise celle spécifiée dans l'appel au constructeur Tween.

Création d'animations s'exécutant indéfiniment

Vous pouvez faire en sorte qu'une animation se poursuive le long de l'axe *x* sans jamais s'arrêter. La classe Tween s'adapte à ce type de demande via la méthode `yoyo()` très justement nommée. La méthode `yoyo()` attend que le gestionnaire d'événements `onMotionFinished` s'exécute, puis inverse les paramètres `begin` et `finish`. L'animation recommence alors, comme dans l'exemple suivant.

Pour créer une animation se poursuivant indéfiniment :

1. Créez un document Flash appelé **yoyo.fla**.
2. Dans le panneau Actions, entrez le code ActionScript suivant sur l'image 1 du scénario :

```
import mx.transitions.Tween;
import mx.transitions.easing.*;

this.createEmptyMovieClip("box_mc", this.getNextHighestDepth());
with (box_mc) {
  beginFill(0xFF0000, 60);
  moveTo(0, 0);
  lineTo(20, 0);
  lineTo(20, Stage.height);
  lineTo(0, Stage.height);
  lineTo(0, 0);
  endFill();
}
```

La première section de code commence par importer la classe Tween, ainsi que chaque classe du paquet d'accélération. La section de code suivante crée un clip en lui donnant le nom d'occurrence `box_mc` et dessine un rectangle d'une largeur de 20 pixels et de même hauteur que la scène.

3. Ajoutez le code ActionScript suivant sous le code créé à l'étape précédente :

```
var box_tween:Tween = new Tween(box_mc, "_x", Regular.easeInOut, 0,
  Stage.width, 3, true);
box_tween.onMotionFinished = function() {
     box_tween.yoyo();
};
```

Ce code crée une interpolation qui anime le clip `box_mc` sur la scène le long de l'axe *x*-pendant 3 secondes.

4. Choisissez Contrôle > Tester l'animation pour tester le clip.

Le carré s'anime de gauche à droite, puis revient. Si l'animation n'est pas suffisamment fluide, vous pouvez augmenter la cadence du document de 12 à 24 ips.

Lorsque le carré s'approche du bord droit de la scène, il sort de son cadre. Bien que cela ne présente guère d'importance, vous préférez peut-être qu'il ne disparaisse pas de la scène pour réapparaître une seconde plus tard et s'animer dans l'autre sens.

Pour effectuer des ajustements, animez le carré de 0 pixel à la largeur de la scène, moins la largeur du clip `box_mc`.

5. Pour que le carré ne disparaisse plus, remaniez le code de l'étape 3 de manière à ce qu'il corresponde au code suivant :

```
var box_tween:Tween = new Tween(box_mc, "_x", Regular.easeInOut, 0,
  (Stage.width - box_mc._width), 3, true);
```

6. Testez à nouveau l'animation (Contrôle > Tester l'animation).

Maintenant, le carré n'accélère plus avant de sortir des limites de la scène.

Association des classes TransitionManager et Tween

D'intéressants effets peuvent être obtenus en combinant les classes TransitionManager et Tween. Vous pouvez utiliser la classe TransitionManager pour déplacer un clip le long de l'axe *x* tout en ajustant la propriété `_alpha` de ce même clip à l'aide de la classe Tween. Chaque classe pouvant utiliser une méthode d'accélération différente, les possibilités d'animation des objets dans vos fichiers SWF sont nombreuses. Vous pouvez profiter des méthodes `continueTo()` et `yoyo()` de la classe Tween ou du gestionnaire d'événements `onMotionFinished` pour créer un effet véritablement unique.

Combinez les classes TransitionManager et Tween pour animer un clip chargé dynamiquement et lui appliquer un fondu sur la scène à la fin de son chargement sur le serveur distant, comme dans l'exemple suivant.

Pour combiner les classes TransitionManager et Tween :

1. Créez un document Flash dénommé **combination.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
import mx.transitions.*;
import mx.transitions.easing.*;

var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function(target_mc:MovieClip) {
    new Tween(target_mc, "_alpha", Strong.easeIn, 0, 100, 2, true);
    TransitionManager.start(target_mc, {type:Fly,
  direction:Transition.IN, duration:3, easing:Elastic.easeInOut,
  startPoint:6});
};

var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.addListener(mcl_obj);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  this.createEmptyMovieClip("img_mc", this.getNextHighestDepth()));
```

   Ce code présente trois sections différentes.

   La première section de code importe les classes dans le paquet de transitions, ainsi que le paquet `transitions.easing`. Dans cet exemple, vous importez la totalité du paquet de transitions. Vous n'avez donc pas besoin d'indiquer le nom pleinement qualifié des classes Tween et TransitionManager ou de la transition sélectionnée (ici, *Fly*). Vous avez ainsi moins de code à rédiger et réduisez le risque d'erreur typographique.

   Le seconde section du code ActionScript crée un objet écouteur pour l'occurrence de la classe MovieClipLoader, créée dans la troisième section du code. Lorsque le clip est chargé dans l'occurrence MovieClipLoader, l'événement `onLoadInit` se déclenche et exécute le bloc de code qui appelle les deux classes Tween et TransitionManager. Ce gestionnaire d'événements applique un fondu dans le clip cible, car vous modifiez la propriété `_alpha` dans la classe Tween, et fait « voler » le clip le long de l'axe *x*.

   La troisième section du code ActionScript crée une occurrence MovieClipLoader et applique l'objet écouteur précédemment créé (afin que l'occurrence du chargeur du clip cible puisse écouter l'événement `onLoadInit`). Vous chargez ensuite l'image JPEG cible dans un clip que vous créez dynamiquement en appelant la méthode `createEmptyMovieClip()`.

3. Enregistrez votre document, puis choisissez Contrôle > Tester l'animation pour tester l'animation dans l'environnement de test.

   Lorsque le téléchargement de l'image JPEG à partir du serveur est terminé, l'image se fond progressivement et s'anime de droite à gauche à travers la scène.

Pour plus d'informations sur l'utilisation de la classe Tween, consultez « Utilisation de la classe Tween », à la page 483.

Pour plus d'informations sur les méthodes et propriétés de la classe Tween et de la classe TransitionManager, consultez la *Référence du langage des composants ActionScript 2.0*.

Pour un exemple de fichier source tweenProgress.fla qui ajoute de l'animation scriptée à l'aide des classes Tween et TransitionManager, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Tween ProgressBar afin d'accéder à cet exemple.

Utilisation d'effets de filtre

Les filtres sont des effets visuels que vous pouvez appliquer à des objets rendus à l'exécution par Flash Player, par exemple des occurrences de clip. Ces filtres permettent ainsi d'appliquer des effets d'ombre, de flou, de rayonnement, de biseau, de rayonnement dégradé et de biseau dégradé. Vous pouvez également utiliser un filtre de réglage des couleurs pour modifier la luminosité, le contraste, la saturation ou les teintes d'un clip. Vous pouvez appliquer des filtres à l'aide de l'interface utilisateur de Flash, dans Flash Professional 8, ou du code ActionScript dans Flash Basic 8 ou Flash Professional 8.

Vous pouvez appliquer chacun de ces effets de filtre à des clips, des boutons ou des champs de texte, soit à l'aide de l'onglet Filtres de l'inspecteur des propriétés, soit à l'aide de code ActionScript. Si vous utilisez du code ActionScript pour appliquer des filtres à une occurrence, vous pouvez aussi utiliser un filtre de mappage du déplacement (voir « Utilisation du filtre mappage de déplacement », à la page 523) ou un filtre de convolution (voir « Utilisation du filtre convolution », à la page 522). Ces filtres sont appliqués aux définitions vectorielles, de sorte que vous n'avez pas à stocker une image bitmap dans un fichier SWF. Vous pouvez également écrire un code ActionScript pour modifier un filtre existant déjà appliqué à un champ de texte, à un clip ou à un bouton.

L'exemple suivant explique comment utiliser un gestionnaire d'événements `onEnterFrame` pour appliquer un filtre de rayonnement à un clip.

Animation d'un effet de filtre appliqué à une occurrence de clip

1. Créez un document Flash dénommé **animFilter.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
this.createEmptyMovieClip("box_mc", 10);
box_mc.lineStyle(20, 0x000000);
box_mc.beginFill(0x000000);
box_mc.moveTo(0, 0);
box_mc.lineTo(160, 0);
box_mc.lineTo(160, 120);
box_mc.lineTo(0, 120);
box_mc.lineTo(0, 0);
box_mc.endFill();
box_mc._x = 100;
box_mc._y = 100;

box_mc.filters = [new flash.filters.GlowFilter()];
var dir:Number = 1;
box_mc.blur = 10;
box_mc.onEnterFrame = function() {
  box_mc.blur += dir;
  if ((box_mc.blur >= 30) || (box_mc.blur <= 10)) {
    dir *= -1;
  }
  var filter_array:Array = box_mc.filters;
  filter_array[0].blurX = box_mc.blur;
  filter_array[0].blurY = box_mc.blur;
  box_mc.filters = filter_array;
};
```

   Ce code possède deux fonctionnalités distinctes. La première section crée et positionne une occurrence de clip et dessine sur la scène un rectangle noir arrondi. Le second bloc de code applique un filtre de rayonnement au rectangle sur scène et définit un gestionnaire d'événements `onEnterFrame` chargé d'animer l'effet de filtre. Le gestionnaire d'événements `onEnterFrame` anime le filtre de rayonnement en lui appliquant un flou compris entre 10 et 30 pixels. Une fois que l'animation atteint une valeur supérieure ou égale à 30 pixels, ou inférieure ou égale à 10 pixels, elle s'anime en sens inverse.
3. Enregistrez les modifications apportées au document Flash, puis choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

Pour plus d'informations sur l'utilisation des filtres dans une application, consultez les sections suivantes :

- « Fonctionnement des paquets de filtres », à la page 493
- « Fonctionnement des filtres, de la mise en cache et de la classe MovieClip », à la page 495
- « A propos de la détection des clics et des filtres de rotation, d'inclinaison et de redimensionnement », à la page 496

- « Application de filtres à des occurrences d'objet et à des occurrences BitmapData », à la page 497
- « A propos du traitement des erreurs, des performances et des filtres », à la page 497

Pour un exemple sur l'utilisation du code ActionScript, Filters.fla, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Filters afin d'accéder à cet exemple.

Fonctionnement des paquets de filtres

Les paquets sont des répertoires qui contiennent un ou plusieurs fichier(s) de classe et résident dans un répertoire de chemin de classe désigné. Par exemple, le package flash.filters est un répertoire sur votre disque dur contenant plusieurs fichiers de classe pour chaque type de filtre (tels que BevelFilter, BlurFilter, DropShadowFilter, etc.) dans Flash. Lorsque les fichiers de classe sont organisés ainsi, vous accédez aux classes de manière spécifique. Soit vous utilisez l'instruction *import* pour importer la classe, soit vous y faites référence à l'aide d'un *nom pleinement qualifié*.

REMARQUE

Pour utiliser l'instruction d'importation, vous devez spécifier ActionScript 2.0 et Flash Player 6 ou une version plus récente dans l'onglet Flash de la boîte de dialogue Paramètres de publication de votre fichier FLA.

L'instruction `import` vous permet d'accéder à des classes sans spécifier leur nom complet. Par exemple, pour utiliser BlurFilter dans un script, vous devez y faire référence par son nom pleinement qualifié (flash.filtres.BlurFilter) ou l'importer ; si vous choisissez de l'importer, vous pouvez y faire référence en spécifiant seulement le nom de sa classe (BlurFilter) dans votre code. Le code ActionScript suivant montre les différences entre l'instruction d'importation et l'emploi de noms pleinement qualifiés de classes.

Si vous choisissez de ne pas importer la classe BlurFilter, votre code doit utiliser le nom pleinement qualifié de la classe (nom du paquet suivi du nom de la classe) pour pouvoir appliquer le filtre :

```
// sans importation
var myBlur:flash.filters.BlurFilter = new flash.filters.BlurFilter(10, 10, 3);
```

Le même code, rédigé sans instruction `import`, vous permet d'accéder à BlurFilter avec le même nom de classe, vous évitant ainsi d'avoir à y faire continuellement référence par le nom complet. Vous avez ainsi moins de code à rédiger et réduisez le risque d'erreur typographique.

```
// avec importation
import flash.filters.BlurFilter;
var myBlur:BlurFilter = new BlurFilter(10, 10, 3);
```

Pour importer plusieurs classes dans un paquet (BlurFilter, DropShadowFilter et GlowFilter, par exemple), vous avez deux manières d'importer chacune des classes. La première manière d'importer de multiples classes consiste à importer chacune des classes à l'aide d'une instruction `import` distincte, comme dans l'exemple suivant :

```
import flash.filters.BlurFilter;
import flash.filters.DropShadowFilter;
import flash.filters.GlowFilter;
```

Toutefois, l'utilisation d'instructions individuelles pour importer plusieurs classes d'un même paquet peut se révéler fastidieuse et augmente le risque d'erreurs typographiques. Vous pouvez éviter l'importation de fichiers de classe individuels en utilisant un *caractère générique*, qui importe toutes les classes d'un certain niveau dans un même paquet. L'exemple de code ActionScript suivant montre comment importer avec un caractère générique :

```
import flash.filters.*;
  // importe chaque classe dans le package flash.filters
```

L'instruction `import` s'applique uniquement au script courant (image ou objet) dans lequel elle est appelée. Par exemple, supposons que vous deviez importer l'ensemble des classes du paquet `macr.util` dans l'image 1 d'un document Flash. Dans cette image, vous pouvez faire référence à des classes dans le paquet en utilisant simplement leur nom de classe, au lieu de leur nom pleinement qualifié. Toutefois, pour utiliser le nom de classe dans un autre script d'image, vous devez faire référence aux classes de ce paquet par leur nom pleinement qualifié ou ajouter une instruction d'importation à l'image qui importe les classes dans ce paquet.

Lorsque vous utilisez des instructions d'importation, les seules classes importées sont celles du niveau que vous spécifiez. Par exemple, si vous importez toutes les classes du paquet `mx.transitions`, seules les classes du répertoire /transitions/ sont importées sans celles des sous-répertoires (les classes du paquet `mx.transitions.easing`, par exemple).

CONSEIL

Si vous importez une classe, mais ne l'utilisez pas dans votre script, cette dernière n'est pas exportée avec le fichier SWF. Vous avez ainsi la possibilité d'importer des paquets volumineux sans vous soucier de la taille du fichier SWF. Le pseudo-code binaire associé à une classe n'est inclus dans un fichier SWF que si cette classe est véritablement utilisée.

Pour un exemple sur l'utilisation du code ActionScript, Filters.fla, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Filters afin d'accéder à cet exemple.

Fonctionnement des filtres, de la mise en cache et de la classe MovieClip

Lors du chargement du fichier SWF, si un clip est associé à un filtre, ce clip se place en mémoire cache en tant que bitmap transparent. Tant qu'au moins un filtre est appliqué au clip, Flash Player met en cache le clip sous forme de bitmap à l'exécution en définissant d'office la propriété `cacheAsBitmap` sur `true`. Le bitmap en cache est alors utilisé comme image source pour les effets de filtre. Tout clip comporte généralement deux bitmaps : le premier bitmap correspond au clip d'origine sans filtre et le second à l'image finale après filtrage. Si vous ne modifiez pas l'apparence d'un clip à l'exécution, l'image finale n'a pas à être mise à jour, ce qui en améliore les performances.

Vous pouvez accéder aux filtres appliqués à une occurrence en appelant la propriété `MovieClip.filters`. L'appel de cette propriété vous renvoie un tableau contenant tous les objets de filtre présentement associés à l'occurrence de clip. Tout filtre est assorti d'un jeu de propriétés spécifique, par exemple :

```
trace(my_mc.filters[0].angle); // 45.0
trace(my_mc.filters[0].distance); // 4
```

Pour accéder aux filtres et les modifier, procédez comme avec un objet de tableau normal. En définissant et en lisant des filtres par la propriété, vous obtenez un *double* de l'objet de filtres, et non pas la *référence*.

Pour modifier un filtre existant, vous pouvez utiliser un code analogue à celui de l'exemple suivant.

Modification des propriétés d'un filtre appliqué à une occurrence de clip

1. Créez un document FLA dénommé **modifyFilter.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
this.createEmptyMovieClip("my_mc", 10);
// dessiner un carré
with (my_mc) {
  beginFill(0xFF0000, 100);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 100);
  lineTo(0, 100);
  lineTo(0, 0);
  endFill();
}
my_mc._x = 100;
my_mc._y = 100;
```

```
// utiliser les valeurs DropShadowFilter par défaut
my_mc.filters = [new flash.filters.DropShadowFilter()];
trace(my_mc.filters[0].distance); // 4
var filter_array:Array = my_mc.filters;
filter_array[0].distance = 10;
my_mc.filters = filter_array;
trace(my_mc.filters[0].distance); // 10
```

La première section de code utilise l'API de dessin pour créer un carré rouge, puis positionne la forme sur la scène. La seconde section du code applique un filtre d'ombre portée au carré. Puis le code crée un tableau provisoire avec les filtres à appliquer au carré rouge sur la scène. La propriété `distance` du premier filtre est définie sur 10 pixels et le filtre modifié est réappliqué à l'occurrence de clip `my_mc`.

3. Sélectionnez Contrôle > Tester l'animation pour tester le document.

REMARQUE

A l'heure actuelle, la rotation des filtres en fonction de la rotation du parent ou de toute autre rotation n'est pas prise en charge. Le filtre de flou applique toujours un effet de flou parfait à l'horizontale ou à la verticale, indépendamment de la rotation ou de l'inclinaison des éléments de l'arborescence des objets parents.

CONSEIL

Le contenu filtré est assorti des mêmes contraintes en termes de taille qu'un contenu ayant la propriété `cacheAsBitmap` définie sur `true`. Si l'auteur zoome exagérément sur le fichier SWF, les filtres ne sont plus visibles dès lors que la représentation bitmap est supérieure à 2 880 pixels dans un sens ou dans l'autre. Lorsque vous publiez des fichiers SWF avec des filtres, il peut être judicieux de désactiver les options du menu Zoom.

Pour un exemple sur l'utilisation du code ActionScript pour appliquer des filtres, Filters.fla, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Filters afin d'accéder à cet exemple.

A propos de la détection des clics et des filtres de rotation, d'inclinaison et de redimensionnement

Les zones filtrées (ombres portées, par exemple) situées hors du cadre de délimitation d'une occurrence de clip ne font pas partie de la surface aux fins de la détection des clics (lorsqu'il s'agit de déterminer si une occurrence chevauche ou touche une autre occurrence).
La détection des clics étant de type vectoriel, il est impossible d'en pratiquer une sur le résultat bitmap. Par exemple, si vous appliquez un filtre biseau à une occurrence de bouton, la détection des clics n'est pas possible sur la fraction biseautée de l'occurrence.

Le redimensionnement, la rotation et l'inclinaison ne sont pas pris en charge par les filtres ; si l'occurrence est elle-même redimensionnée (`_xscale` et `_yscale` non définis sur 100 %), l'effet de filtre ne s'ajuste pas à l'occurrence. Cela signifie que la forme originale de l'occurrence est certes pivotée, inclinée ou redimensionnée, mais que le filtre ne pivote pas, ne s'incline pas et ne s'ajuste pas à l'occurrence.

Vous pouvez animer une occurrence avec un filtre afin de créer des effets réalistes, tout comme vous pouvez imbriquer des occurrences et utiliser la classe BitmapData pour animer des filtres afin d'atteindre ces effets.

Application de filtres à des occurrences d'objet et à des occurrences BitmapData

L'utilisation de filtres dépend de l'occurrence d'objet à laquelle vous appliquez le filtre. Suivez les instructions ci-après pour appliquer un filtre à une occurrence d'objet ou à une occurrence BitmapData :

- Pour appliquer des filtres à des clips, à des champs de texte et à des boutons à l'exécution, utilisez la propriété `filters`. Lorsque vous définissez la propriété `filters` d'un objet, celui-ci n'est pas modifié. En outre, vous pouvez l'annuler en effaçant la propriété `filters`.
- Pour appliquer des filtres à des occurrences BitmapData, utilisez la méthode `BitmapData.applyFilter()`. L'appel de `applyFilter()` pour un objet BitmapData génère une image filtrée à partir de l'objet BitmapData source et de l'objet filtre.

REMARQUE	Vous pouvez également appliquer des effets de filtre à des images et à des vidéos pendant la programmation à l'aide de l'onglet Filtres de l'inspecteur des propriétés.

A propos du traitement des erreurs, des performances et des filtres

Si vous utilisez trop de filtres dans une application, vous risquez de consommer beaucoup de mémoire et d'amoindrir les performances de Flash Player. Un clip associé à des filtres présente en effet deux bitmaps de chacun 32 bits. Or, plus vous utilisez de bitmaps, plus votre application consomme de mémoire. Le système d'exploitation de l'ordinateur risque alors de générer une erreur de type saturation de la mémoire. Ce type d'erreur est toutefois rare sur les ordinateurs modernes, à moins que vous n'utilisiez fréquemment des effets de filtre dans une application (par exemple, lorsque vous avez plusieurs milliers de bitmaps sur la scène).

Si, malgré tout, vous rencontrez une erreur de type saturation de la mémoire, voici ce qui se passe :

- Le tableau de filtres est ignoré.
- Le clip est tracé au moyen de la fonctionnalité de rendu vectoriel standard.
- Aucun bitmap n'est mis en cache pour le clip.

Dès lors qu'apparaît une erreur de type saturation de la mémoire, le clip n'essaie plus d'utiliser un tableau de filtres, ni même un cache de bitmaps. Un autre facteur risque d'amoindrir les performances de lecture : la valeur que vous utilisez pour le paramètre `qualité` de chacun des filtres que vous appliquez. Des valeurs élevées exigent davantage de mémoire et de ressources processeur pour l'effet à rendre, tandis que la définition du paramètre `quality` à une valeur moindre demande moins de ressources de la part de l'ordinateur. Il est donc conseillé de ne pas employer trop de filtres et de définir une `qualité` aussi basse que possible.

ATTENTION

Si un zoom avant est pratiqué une fois sur un objet de 100 x 100 pixels, la mémoire est utilisée quatre fois, puisque les dimensions du contenu sont alors de 200 x 200 pixels. Si vous zoomez encore deux fois, la forme est tracée aux dimensions de 800 x 800 pixels. Compte tenu des dimensions d'origine de l'objet (100 x 100 pixels), la mémoire est alors utilisée 64 fois. Ainsi, pour utiliser des filtres dans un fichier SWF, il est recommandé de désactiver les options du menu Zoom à partir du menu contextuel du fichier SWF.

Vous risquez également de rencontrer des erreurs si vous utilisez des types de paramètres non valides. Certains paramètres de filtre ont aussi une plage valide spécifique. Si vous définissez une valeur hors de cette plage valide, le paramètre adopte automatiquement une valeur valide située dans cette plage. Par exemple, le paramètre `qualité` doit présenter une valeur comprise entre 1 et 3 pour un fonctionnement normal et ne peut être défini que de 0 à 15. Toute définition supérieure à 15 revient automatiquement sur 15.

De même, certains constructeurs imposent des restrictions à la longueur des tableaux pour les paramètres d'entrée. Si un filtre de convolution ou un filtre matrice de couleurs est créé avec un tableau non valide (longueur erronée), le constructeur échoue et le filtre n'est pas correctement créé. Si l'objet de filtre est ensuite utilisé en entrée dans un tableau de filtres de clip, il est tout simplement ignoré.

CONSEIL

Lorsque vous utilisez un filtre de flou, il est préférable d'utiliser pour blurX et blurY des valeurs qui sont des puissances de 2 (2, 4, 8, 16 et 32, par exemple), car elles se calculent plus rapidement et améliorent les performances de 20 à 30 %.

Utilisation des filtres avec ActionScript

Le package flash.filters contient des classes pour les effets de filtrage de bitmaps dans Flash Player 8 et les versions ultérieures. Les filtres permettent d'utiliser ActionScript pour appliquer des effets visuels riches, tels que les effets de flou, biseau, rayonnement et ombres portées à des occurrences de texte, de clip et de bouton. Vous pouvez aussi utiliser l'outil de programmation Flash pour appliquer des effets de filtre à des objets, par exemple des textes, des images et des vidéos. Flash possède neuf effets de filtre, mais seulement sept sont accessibles via l'interface utilisateur. Les filtres ConvolutionFilter et DisplacementMapFilter ne sont disponibles qu'avec le code ActionScript.

REMARQUE	Tous les filtres sont disponibles avec ActionScript

Dans l'exemple suivant, une image PNG semi-transparente est chargée et un effet GlowFilter est appliqué à la partie non transparente de l'image.

Application de filtres à des images semi-transparentes

1. Créez un document Flash dénommé **transparentImg.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
import flash.filters.GlowFilter;
System.security.allowDomain("http://www.helpexamples.com");
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
  var glow:GlowFilter = new GlowFilter();
  target_mc.filters = [glow];
};
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/logo.png",
  img_mc);
```

   Ce code utilise une occurrence de chargeur de clip pour charger une image PNG semi-transparente. Une fois chargée, l'image se déplace au centre de la scène et un filtre de rayonnement lui est appliqué.

3. Choisissez Contrôle > Tester l'animation pour tester le document.

   L'effet de filtre de rayonnement n'est appliqué qu'à la zone opaque (non transparente) de l'image PNG.

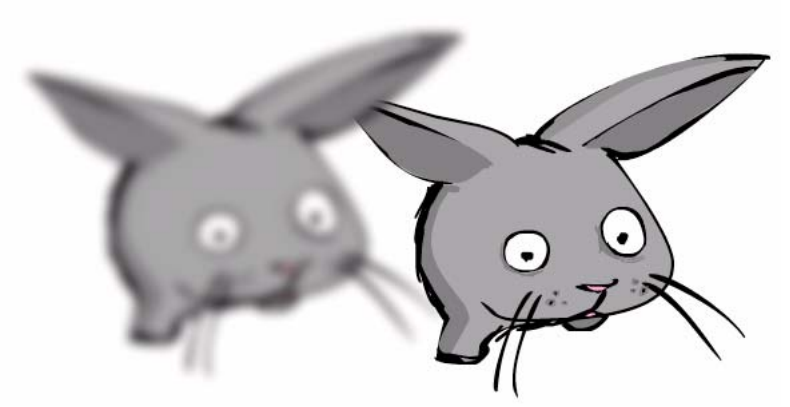
Les sections suivantes expliquent comment utiliser les filtres :

- « Utilisation du filtre de flou », à la page 501
- « Utilisation du filtre d'ombre portée », à la page 503
- « Utilisation du filtre de rayonnement », à la page 508
- « Création de rayonnements dégradés », à la page 509
- « Utilisation du filtre de biseau », à la page 511
- « Application d'un filtre de biseau dégradé », à la page 518
- « Utilisation du filtre matrice de couleurs », à la page 519
- « Utilisation du filtre convolution », à la page 522
- « Utilisation du filtre mappage de déplacement », à la page 523

Pour un exemple sur l'utilisation du code ActionScript pour appliquer des filtres, Filters.fla, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Filters afin d'accéder à cet exemple.

Utilisation du filtre de flou

La classe BlurFilter vous permet d'appliquer un effet visuel de flou à divers objets dans Flash. Un effet de flou adoucit les détails d'une image. Vous pouvez produire une panoplie de flous vous permettant d'obtenir un aspect doux sans focus, un flou gaussian ou encore un aspect voilé, dont l'effet est identique à celui d'une image que l'on regarde à travers un verre semi-opaque. Le filtre de flou est dérivé d'un filtre de flou emboîté. Le paramètre `quality` détermine le nombre de répétitions du flou (trois passages ont l'effet d'un filtre de flou gaussien).



REMARQUE	Le filtre de flou ne s'ajuste que lorsque vous zoomez sur la scène.

Pour plus d'informations sur ce filtre, consultez la section BlurFilter (flash.filters.BlurFilter) dans le *Guide de référence du langage ActionScript 2.0*.

Dans l'exemple suivant, un flou est appliqué à une image chargée dynamiquement à partir de la position courante du pointeur de la souris sur la scène. Plus le pointeur est éloigné du centre de la scène, plus l'image est floue.

Application d'un flou à une image à partir de la position du pointeur de la souris

1. Créez un document Flash dénommé **dynamicblur.fla**.
2. Ajoutez le code suivant à l'image 1 du scénario :

```
import flash.filters.BlurFilter;
System.security.allowDomain("http://www.helpexamples.com");
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  // Centrer le clip target_mc sur la scène
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
};
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);
var blur:BlurFilter = new BlurFilter(10, 10, 2);

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  /* Déplacer le pointeur vers le centre de la scène définit les
  propriétés blurX et blurY sur 0 %. */
  blur.blurX = Math.abs(_xmouse - (Stage.width / 2)) / Stage.width * 2 *
  255;
  blur.blurY = Math.abs(_ymouse - (Stage.height / 2)) / Stage.height * 2
  * 255;
  img_mc.filters = [blur];
};
Mouse.addListener(mouseListener);
```

La première section de ce code charge et positionne une image chargée dynamiquement sur la scène. La seconde section définit un écouteur, qui est appelé à chaque fois que la souris bouge. Le montant de flou horizontal et vertical se calcule en fonction de la position courante du pointeur de la souris sur la scène. Plus vous éloignez le pointeur du centre de la scène, plus le montant du flou appliqué à l'occurrence est élevé.

3. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

   Déplacez le pointeur de la souris le long de l'axe *x* pour modifier le montant de flou horizontal. L'occurrence devient plus floue à mesure que le pointeur s'éloigne du centre horizontal de la scène. En déplaçant le pointeur le long de l'axe *y*, le flou vertical augmente ou diminue, en fonction de la distance à partir du centre vertical de la scène.

CONSEIL	Lorsque vous utilisez un filtre de flou, il est préférable d'utiliser pour blurX et blurY des valeurs qui sont des puissances de 2 (2, 4, 8, 16 et 32, par exemple), car elles se calculent plus rapidement et améliorent les performances de 20 à 30 %.

ATTENTION	Une valeur de flou inférieure à 1,03125 désactive l'effet de flou.

Utilisation du filtre d'ombre portée

La classe DropShadowFilter vous permet d'ajouter une ombre portée à divers objets dans Flash. L'algorithme d'ombre est dérivé du même filtre que celui utilisé par le filtre de flou (voir « Utilisation du filtre de flou », à la page 501). Vous disposez de plusieurs options pour définir le style de l'ombre portée, notamment l'ombre intérieure ou extérieure et le mode de masquage.

Pour plus d'informations sur ce filtre, consultez la section DropShadowFilter (flash.filters.DropShadowFilter) dans le *Guide de référence du langage ActionScript 2.0*.

Dans l'exemple suivant, un carré est tracé sur la scène à l'aide de l'API de dessin. Lorsque vous déplacez horizontalement le pointeur de la souris sur la scène, ce code modifie la distance à laquelle apparaît l'ombre portée à partir du carré ; lorsque vous déplacez le pointeur verticalement, c'est le montant du flou de l'ombre portée qui est modifié.

Utilisation du filtre d'ombre portée :

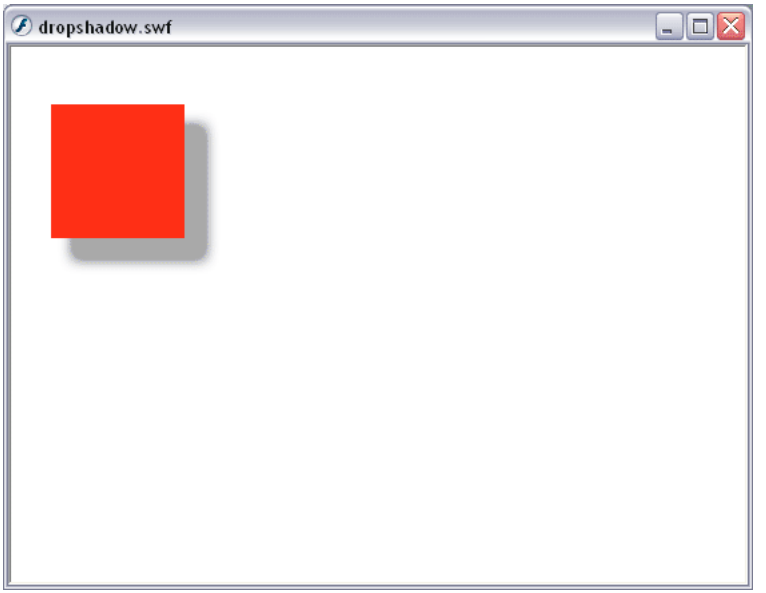
1. Créez un document Flash dénommé **dropshadow.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
// Importation des classes de filtre
import flash.filters.DropShadowFilter;
// créer un clip dénommé shapeClip
this.createEmptyMovieClip("shapeClip", 1);
// utiliser l'API de dessin pour tracer une forme
with (shapeClip) {
  beginFill(0xFF0000, 100);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 100);
  lineTo(0, 100);
  lineTo(0, 0);
  endFill();
}
// positionner la forme
shapeClip._x = 100;
shapeClip._y = 100;
// cliquer sur le carré, augmenter l'intensité de l'ombre
shapeClip.onPress = function():Void {
  dropShadow.strength++;
  shapeClip.filters = [dropShadow];
};
// créer un filtre
var dropShadow:DropShadowFilter = new DropShadowFilter(4, 45, 0x000000,
  0.4, 10, 10, 2, 3);

var mouseListener:Object = new Object();
// créer et appliquer un écouteur pour contrôler le filtre suivant
// les déplacements de la souris
mouseListener.onMouseMove = function():Void {
  dropShadow.distance = (_xmouse / Stage.width) * 50 - 20;
  dropShadow.blurX = (_ymouse / Stage.height) * 10;
  dropShadow.blurY = dropShadow.blurX;
  shapeClip.filters = [dropShadow];
};
Mouse.addListener(mouseListener);
```

La première section de code crée un clip et utilise l'API de dessin pour tracer un carré rouge. La seconde section définit un écouteur de souris, qui est appelé à chaque fois que la souris bouge. L'écouteur de souris calcule la distance de l'ombre portée et le niveau de flou sur la base de la position courante, sur les axes *x* et *y*, du pointeur de la souris, puis applique à nouveau le filtre d'ombre portée. Si vous cliquez sur le carré rouge, l'intensité de l'ombre portée augmente.

3. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

   Déplacez le pointeur de la souris le long de l'axe *x* pour modifier la valeur de distance de l'ombre portée, puis déplacez le pointeur de la souris le long de l'axe *y* pour modifier le montant de flou appliqué à l'occurrence de clip.



Vous pouvez également créer des ombres portées et les appliquer à des images chargées dynamiquement. L'exemple suivant vous explique comment charger une image externe et lui appliquer une ombre portée suivant le pointeur de la souris. Plus le pointeur s'éloigne du coin supérieur gauche de l'image, plus le montant de flou horizontal et vertical appliqué à l'image est élevé.

Création d'une ombre portée suivant le pointeur de la souris

1. Créez un document Flash et enregistrez-le sous le nom de **dropshadowmouse.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
import flash.filters.DropShadowFilter;
System.security.allowDomain("http://www.helpexamples.com");
var dropShadow:DropShadowFilter = new DropShadowFilter(4, 45, 0x000000,
  0.8, 10, 10, 2, 2);
// charger et positionner l'image sur la scène
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip):Void {
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
};
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);

// lorsque la souris est déplacée, recalculer la position de
// l'ombre portée
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  var p1:Number = img_mc._y - _ymouse;
  var p2:Number = img_mc._x - _xmouse;
  var degrees:Number = Math.atan2(p1, p2) / (Math.PI / 180);
  dropShadow.distance = Math.sqrt(Math.pow(p1, 2) + Math.pow(p2, 2)) *
  0.5;
  dropShadow.blurX = dropShadow.distance;
  dropShadow.blurY = dropShadow.blurX;
  dropShadow.angle = degrees - 180;
  img_mc.filters = [dropShadow];
};
Mouse.addListener(mouseListener);
```

La première section de ce code définit une occurrence d'ombre portée, charge une image externe et repositionne l'image au centre de la scène. La seconde section définit un écouteur de souris, qui est appelé à chaque fois que l'utilisateur déplace le pointeur de la souris sur la scène. A chaque fois que la souris est déplacée, le gestionnaire d'événements recalcule la distance et l'angle entre le pointeur de la souris et le coin supérieur gauche de l'image. Sur la base de ces calculs, le filtre d'ombre portée est réappliqué au clip.

3. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

   Lorsque vous exécutez le fichier SWF, l'ombre portée suit le pointeur de la souris. Plus le pointeur de la souris est proche du coin supérieur gauche de l'image sur la scène, moins l'effet de flou appliqué à l'image est important. L'effet d'ombre portée devient plus apparent à mesure que le pointeur de la souris s'éloigne du coin supérieur gauche de l'image.

Vous pouvez également appliquer des ombres portées à des images PNG semi-transparentes chargées dynamiquement. Dans l'exemple suivant, le filtre d'ombre portée est uniquement appliqué à la zone pleine de l'image PNG, et non pas à la zone transparente.

Application d'une ombre portée à une image semi-transparente

1. Créez un document Flash et enregistrez-le sous le nom de **dropshadowTransparent.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
import flash.filters.DropShadowFilter;
System.security.allowDomain("http://www.helpexamples.com");
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip):Void {
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
  var dropShadow:DropShadowFilter = new DropShadowFilter(4, 45,
  0x000000, 0.5, 10, 10, 2, 3);
  target_mc.filters = [dropShadow];
};
mclListener.onLoadError = function(target_mc:MovieClip):Void {
  trace("unable to load image.");
};
this.createEmptyMovieClip("logo_mc", 10);
var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.addListener(mclListener);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/logo.png",
  logo_mc);
```

   Ce code ActionScript utilise la classe MovieClipLoader pour charger une image et appliquer un filtre d'ombre portée une fois que l'image est complètement chargée à partir du serveur distant.

3. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

   Flash charge une image PNG avec un arrière-plan transparent. Lorsque vous appliquez le filtre d'ombre portée, celui-s'applique à la seule partie opaque (non transparente) de l'image.

Utilisation du filtre de rayonnement

La classe GlowFilter permet d'ajouter un effet de rayonnement à divers objets dans Flash. L'algorithme de rayonnement est dérivé du même filtre que celui utilisé par le filtre de flou (voir « Utilisation du filtre de flou », à la page 501). Vous disposez de plusieurs options pour définir le style de rayonnement, notamment le rayonnement interne ou externe et le mode de masquage. Le filtre de rayonnement est analogue au filtre d'ombre portée dont les propriétés `distance` et `angle` sont définies sur `0`.

Pour plus d'informations sur le filtre de rayonnement, consultez la section GlowFilter (flash.filters.GlowFilter) dans le *Guide de référence du langage ActionScript 2.0*.

L'exemple suivant explique comment appliquer un filtre de rayonnement à un clip créé dynamiquement sur la scène. Il suffit de déplacer le pointeur de la souris sur la scène pour modifier le flou du clip et de cliquer sur la forme créée dynamiquement pour augmenter l'intensité du filtre.

Utilisation du filtre de rayonnement

1. Créez un document Flash et enregistrez-le sous le nom de **glowfilter.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
import flash.filters.GlowFilter;

this.createEmptyMovieClip("shapeClip", 10);
with (shapeClip) {
  beginFill(0xFF0000, 100);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 100);
  lineTo(0, 100);
  lineTo(0, 0);
  endFill();
}
shapeClip._x = 100;
shapeClip._y = 100;
shapeClip.onPress = function():Void {
  glow.strength++;
  shapeClip.filters = [glow];
};
var glow:GlowFilter = new GlowFilter(0xCC0000, 0.5, 10, 10, 2, 3);
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  glow.blurX = (_xmouse / Stage.width) * 255;
  glow.blurY = (_ymouse / Stage.width) * 255;
  shapeClip.filters = [glow];
};
Mouse.addListener(mouseListener);
```

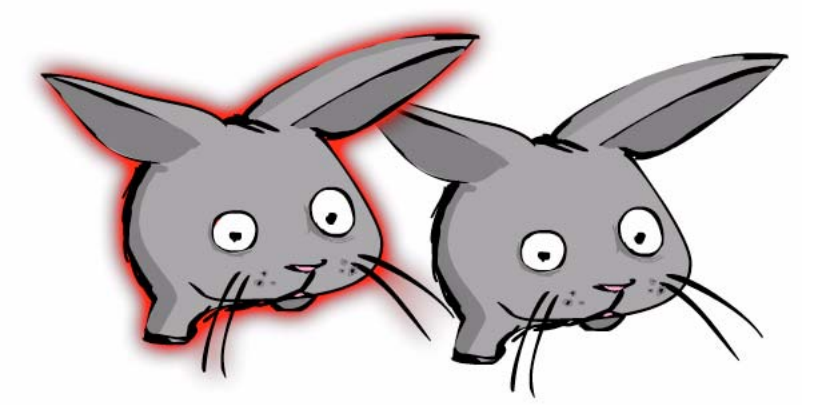
Ce code utilise l'API de dessin pour tracer un carré sur la scène, puis applique un filtre de rayonnement à la forme. Dès que le pointeur de la souris se déplace le long de l'axe *x* ou *y*, le flou du filtre de rayonnement est calculé et appliqué à la forme.

3. Choisissez Contrôle > Tester l'animation pour tester le document.

   Le montant de flou horizontal et vertical se calcule en fonction de la position courante du pointeur de la souris `_xmouse` et `_ymouse`. Le montant de flou horizontal et vertical diminue à mesure que le pointeur de la souris s'approche du coin supérieur gauche de la scène. Inversement, le montant de flou horizontal et vertical augmente à mesure que le pointeur de la souris s'approche du coin supérieur droit de la scène.

Création de rayonnements dégradés

La classe GradientGlowFilter vous permet de créer un effet de rayonnement dégradé pour l'appliquer à divers objets dans Flash. L'aspect d'un rayonnement dégradé est réaliste et produit un dégradé de couleurs que vous pouvez spécifier. Vous pouvez appliquer un rayonnement dégradé autour du bord intérieur ou extérieur d'un objet, ou encore sur son bord supérieur.



Pour plus d'informations sur ce filtre, consultez la section GradientBevelFilter (flash.filters.GradientBevelFilter) dans le *Guide de référence du langage ActionScript 2.0*.

Le code suivant utilise l'API de dessin pour tracer un carré sur la scène, puis applique un filtre de rayonnement dégradé à la forme. Il suffit de cliquer sur le carré sur la scène pour augmenter l'intensité du filtre et de déplacer le pointeur de la souris horizontalement ou verticalement le long de l'axe *x* ou *y* pour modifier le montant de flou.

Application d'un filtre de rayonnement dégradé :

1. Créez un document Flash et enregistrez-le sous le nom de **gradientglow.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
import flash.filters.GradientGlowFilter;
// créer une occurrence shapeClip
var shapeClip:MovieClip = this.createEmptyMovieClip("shapeClip", 10);
// utiliser l'API de dessin pour créer une forme
with (shapeClip) {
  beginFill(0xFF0000, 100);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 100);
  lineTo(0, 100);
  lineTo(0, 0);
  endFill();
}

// positionner la forme
shapeClip._x = 100;
shapeClip._y = 100;
// définir un rayonnement dégradé
var gradientGlow:GradientGlowFilter = new GradientGlowFilter(0, 45,
  [0x000000, 0xFF0000], [0, 1], [0, 255], 10, 10, 2, 3, "outer");

// définir un écouteur de souris, écouter deux événements
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function():Void {
  gradientGlow.strength++;
  shapeClip.filters = [gradientGlow];
};
mouseListener.onMouseMove = function():Void {
  gradientGlow.blurX = (_xmouse / Stage.width) * 255;
  gradientGlow.blurY = (_ymouse / Stage.height) * 255;
  shapeClip.filters = [gradientGlow];
};
Mouse.addListener(mouseListener);
```

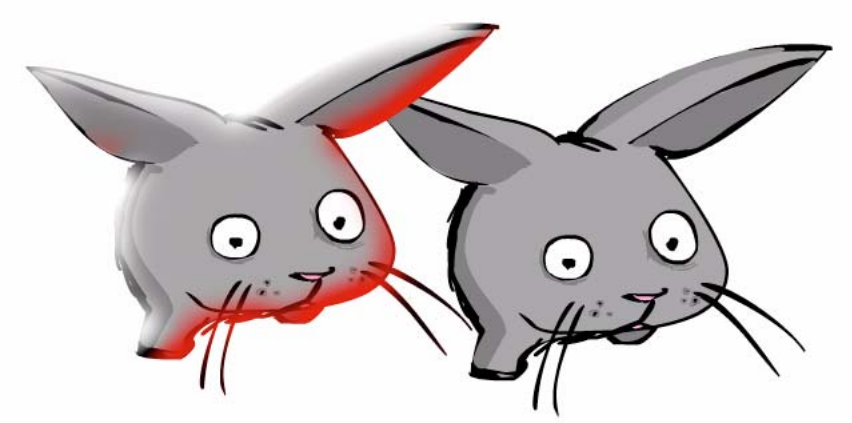
Ce code se compose de trois sections différentes. La première section de code utilise l'API de dessin pour créer un carré, puis positionne la forme sur la scène. La deuxième section de code définit une nouvelle occurrence de filtre de rayonnement dégradé, qui crée un dégradé du rouge vers le noir. La troisième section définit un écouteur de souris, qui écoute deux gestionnaires d'événements de souris. Le premier gestionnaire d'événements est `onMouseDown` qui augmente l'intensité du rayonnement dégradé. Le second gestionnaire d'événements est `onMouseMove` qui est appelé à chaque fois que le pointeur de la souris se déplace dans le fichier SWF. L'effet de rayonnement dégradé devient plus intense à mesure que le pointeur de la souris s'éloigne du coin supérieur gauche du document Flash.

3. Choisissez Contrôle > Tester l'animation pour tester le document.

   Le flou du filtre de rayonnement dégradé augmente ou diminue en intensité à mesure que le pointeur de la souris évolue sur la scène. Cliquez sur le bouton gauche de la souris pour augmenter l'intensité du rayonnement.

Utilisation du filtre de biseau

La classe BevelFilter vous permet d'ajouter un effet de biseau à divers objets dans Flash. L'effet de biseau donne aux objets un aspect tridimensionnel. Vous pouvez personnaliser l'aspect du biseau en modifiant les couleurs de surlignement et d'ombre, le montant de flou sur le biseau, l'angle du biseau, la position du biseau et l'effet de masquage.



Pour plus d'informations sur ce filtre, consultez la section BevelFilter (flash.filters.BevelFilter) dans le *Guide de référence du langage ActionScript 2.0*.

Dans l'exemple suivant, un carré est tracé sur la scène à l'aide de l'API de dessin, puis un biseau est ajouté à la forme.

Utilisation du filtre de biseau :

1. Créez un document Flash et enregistrez-le sous le nom de **bevel.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
import flash.filters.BevelFilter;
// définir un filtre de biseau
var bevel:BevelFilter = new BevelFilter(4, 45, 0xFFFFFF, 1, 0xCC0000, 1,
  10, 10, 2, 3);
// créer une occurrence shapeClip
var shapeClip:MovieClip = this.createEmptyMovieClip("shapeClip", 1);
// utiliser l'API de dessin pour créer une forme
with (shapeClip) {
  beginFill(0xFF0000, 100);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 100);
  lineTo(0, 100);
  lineTo(0, 0);
  endFill();
}
// positionner la forme sur la scène
shapeClip._x = 100;
shapeClip._y = 100;
// cliquer pour augmenter l'intensité
shapeClip.onPress = function():Void {
  bevel.strength += 2;
  shapeClip.filters = [bevel];
};

// définir un écouteur pour modifier le filtre suivant les
// déplacements du pointeur
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  bevel.distance = (_xmouse / Stage.width) * 10;
  bevel.blurX = (_ymouse / Stage.height) * 10;
  bevel.blurY = bevel.blurX;
  shapeClip.filters = [bevel];
};
Mouse.addListener(mouseListener);
```

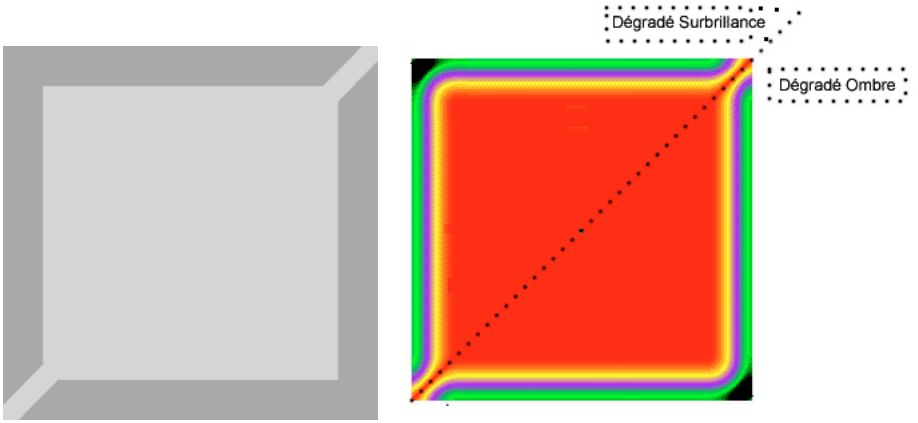
La première section de code crée une occurrence BevelFilter et utilise l'API de dessin pour tracer un carré sur la scène. Lorsque vous cliquez sur le carré sur la scène, la valeur courante de l'intensité du biseau s'incrémente, ce qui fait apparaître le biseau plus grand et plus net. La seconde section de code définit un écouteur de souris, qui modifie le flou et la distance du biseau en fonction de la position courante du pointeur de la souris.

3. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

   Déplacez le pointeur de la souris le long de l'axe *x* pour augmenter ou réduire la distance de décalage du biseau. Lorsque vous déplacez le pointeur de la souris le long de l'axe *y*, les coordonnées courantes du pointeur de la souris modifient le montant de flou horizontal et vertical.

A propos du filtre de biseau dégradé

Le filtre de biseau dégradé s'applique à des objets, par exemple un rectangle, avec répartition des couleurs du dégradé sur trois parties du rectangle : les deux bords du biseau (un *surlignement* et une *ombre*) et une zone que nous appellerons *fond*. Les graphiques suivants représentent le rectangle, avec le type de biseau défini sur « intérieur » : Dans le rectangle de gauche, les zones gris foncé représentent les bords du biseau, tandis que la zone gris clair en représente le fond. Dans le rectangle de droite, un biseau de type arc-en-ciel dégradé, avec un biseau de quatre couleurs sur chaque bord, est appliqué.



Les différentes propriétés du filtre de biseau dégradé contrôlent le mode d'application du filtre. Les couleurs du biseau dégradé sont définies dans le tableau de couleurs. La répartition effective des couleurs dans chaque partie du rectangle est déterminée par le tableau de rapports. La propriété de distance détermine la distance de décalage ou le nombre de pixels appliqué au bord du biseau à distance de l'objet. Les propriétés `blurX` et `blurY` contrôlent la netteté des couleurs dans le biseau ; des valeurs élevées donnent un biseau plus large et plus doux tandis que des valeurs inférieures le rendent plus fin et plus net. La propriété d'angle génère la source lumineuse théorique qui tombe sur l'objet, créant un effet de surlignement et d'ombre sur les bords de l'objet. La propriété `intensité` contrôle l'étalement des couleurs : une valeur d'intensité basse estompe les couleurs, comme dans l'exemple ; une valeur d'intensité élevée rend les nombres des extrémités du tableau plus intenses, obligeant les couleurs situées au milieu de tableau à perdre en intensité. Enfin, les propriétés `masquage` et `type` déterminent où et comment le filtre de biseau est appliqué à l'ensemble de l'objet : si le filtre masque l'objet et où il est placé.

Le plus compliqué, pour appliquer le filtre de biseau dégradé, réside dans la distribution des couleurs. Pour bien comprendre le mode de distribution des couleurs dans un biseau dégradé, vous devez d'abord déterminer les couleurs que vous souhaitez retrouver dans votre biseau. A l'instar du biseau simple, qui peut s'assortir de couleurs de surlignement et de couleurs d'ombre, le filtre de biseau dégradé autorise ces deux possibilités : le dégradé de surlignement et le dégradé d'ombre. Le surlignement apparaît dans le coin supérieur gauche et l'ombre dans le coin inférieur droit. Il y a quatre couleurs disponibles pour le surlignement et quatre aussi pour l'ombre. Toutefois, il vous faut ajouter une autre couleur (couleur de fond), qui apparaîtra aux intersections du surlignement et de l'ombre. Comme vous pouvez le voir dans le diagramme précédent, le tableau de couleurs compte neuf couleurs.

Le nombre de couleurs dans le tableau de couleurs détermine le nombre d'éléments dans le tableau d'alphas et de rapports. Le premier élément du tableau de couleurs correspond au premier élément du tableau d'alphas et du tableau de rapports, et ainsi de suite. Comme vous avez neuf couleurs, vous avez aussi neuf valeurs dans le tableau d'alphas et neuf valeurs dans le tableau de rapports. Les valeurs alpha déterminent la valeur de transparence alpha des couleurs.

Les valeurs des rapports dans le tableau de rapports sont comprises entre 0 et 255 pixel(s). La valeur du milieu est 128 : il s'agit de la valeur de fond. Dans la plupart des cas, pour obtenir l'effet de biseau recherché, il suffit d'affecter les valeurs de rapport comme suit, sur la base d'un exemple de neuf couleurs :

- Les quatre premières couleurs vont de 0 à 127 et augmentent en valeur, de sorte que chaque valeur est toujours égale ou supérieure à la précédente. Il s'agit de la première bordure du biseau, c'est-à-dire le surlignement.

- La cinquième couleur (celle du milieu) correspond au fond, défini sur 128. La valeur de 128 pixels définit donc le fond, qui apparaît soit à l'extérieur de la forme (et autour des bordures du biseau) si le type est défini sur extérieur, soit à l'intérieur de la forme, couvrant toute la zone de remplissage de l'objet, si le type est défini sur intérieur.
- Les quatre dernières couleurs vont de 129 à 255 et augmentent en valeur, de sorte que chaque valeur est toujours égale ou supérieure à la précédente. Il s'agit de la seconde bordure du biseau, c'est-à-dire l'ombre.

Si l'on prend un dégradé composé de rayures de différentes couleurs, mélangées les unes aux autres, chaque valeur de rapport détermine le nombre de pixels pour la couleur associée, ce qui définit la largeur de la rayure de couleur dans le dégradé. Si vous souhaitez une distribution homogène des couleurs pour chaque bordure :

- Définissez un nombre impair de couleurs, où la valeur du milieu correspond au fond.
- Distribuez les valeurs entre 0 et 127 et entre 129 et 255 de manière homogène parmi vos couleurs.
- Ajustez la valeur de manière à modifier la largeur de chaque rayure de couleur dans le dégradé.

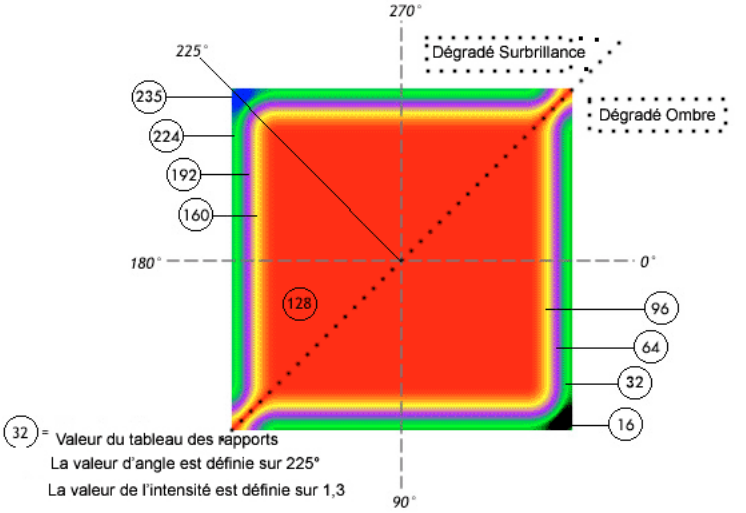
REMARQUE	La valeur d'angle détermine la bordure de surlignement et la bordure d'ombre.

La valeur d'angle détermine l'angle auquel les couleurs dégradées sont appliquées à l'objet, c'est-à-dire là où le surlignement et l'ombre vont apparaître sur l'objet. Les couleurs sont appliquées dans le même ordre que dans le tableau.

Le code suivant prend un carré rose (tracé avec l'API de dessin) et applique un filtre de type arc-en-ciel dégradé. Les couleurs, dans l'ordre où elles sont présentées dans le tableau, sont les suivantes : bleu, vert, violet et jaune (surlignement) ; rouge (fond) ; jaune, violet, vert et noir (ombre). Pour déterminer les valeurs des rapports, nous affectons quatre valeurs de couleurs de surlignement de 0 à 127, en faisant en sorte qu'elles soient plus ou moins égales, et quatre valeurs de couleurs d'ombre de 129 à 255. Les couleurs aux bordures extérieures sont le bleu (16) et le noir (235).

```
var colors:Array = [0x0000FF, 0x00FF00, 0x9900FF, 0xFFFF00, 0xFF0000,
  0xFFFF00, 0x9900FF, 0x00FF00,0x000000];
var alphas:Array = [1, 1, 1, 1, 1, 1, 1, 1, 1];
var ratios:Array = [16, 32, 64, 96, 128, 160, 192, 224, 235];
var gradientBevel:GradientBevelFilter = new GradientBevelFilter(8, 225,
  colors, alphas, ratios, 16, 16, 1.3, 2, "inner", false);
```

La figure suivante montre le filtre de biseau dégradé créé par le code ci-dessus, à savoir un biseau de type arc-en-ciel dégradé en neuf couleurs appliqué à un clip rouge rectangulaire :



La ligne pointillée indique la manière dont les angles sont déterminés. La figure montre la manière dont l'angle de 225° est réalisé sur le filtre et indique la valeur de rapport de chaque couleur. La définition de l'angle à 225° indique que la première couleur dans le tableau débute à 225°, c'est-à-dire dans le coin supérieur gauche (surlignement). La ligne pointillée indique l'endroit où le dégradé de surlignement est appliqué et où le dégradé d'ombre est appliqué.

La couleur d'origine du clip est le rose mais, si l'on définit la valeur 128 sur le rouge, la valeur de 128 pixels devient la couleur de fond et couvre toute la zone de remplissage du clip d'origine. Cependant, lorsque vous définissez la propriété de filtre, l'objet d'origine n'est pas modifié ; si vous effacez la propriété `filtres`, vous pouvez restaurer le remplissage d'origine du clip.

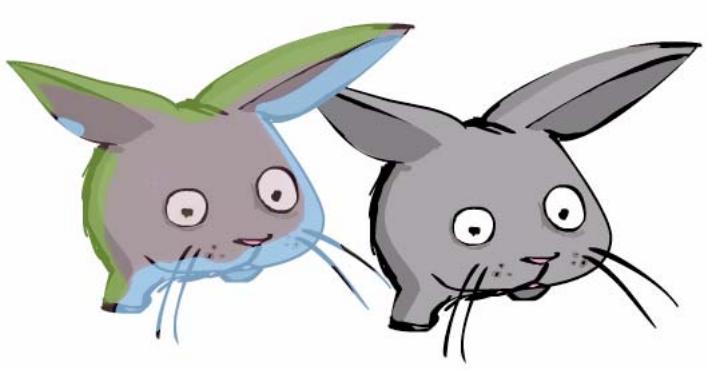
Les propriétés de tous les filtres étant intercorrélées, il se peut que vous ayez à redéfinir certaines propriétés dès lors que vous ajustez une propriété afin de modifier l'effet que vous appliquez.

Le code ActionScript complet utilisé pour créer la figure précédente est le suivant :

```
import flash.filters.GradientBevelFilter;

// tracer une forme de carré remplie
this.createEmptyMovieClip("square_mc", this.getNextHighestDepth());
    square_mc.beginFill(0xFF99CC);
    square_mc.moveTo(40, 40);
    square_mc.lineTo(200, 40);
    square_mc.lineTo(200, 200);
    square_mc.lineTo(40, 200);
    square_mc.lineTo(40, 40);
    square_mc.endFill();

/* GradientBevelFilter(distance:Number, angle:Number, colors:Array,
  alphas:Array, ratios:Array, blurX:Number, blurY:Number, strength:Number,
  quality:Number, type:String, knockout:Boolean) */

// créer des tableaux de couleurs, d'alphas et de rapports
var colors:Array = [0x0000FF, 0x00FF00, 0x9900FF, 0xFFFF00, 0xFF0000,
  0xFFFF00, 0x9900FF, 0x00FF00,0x000000];
  //bleu, vert, violet, jaune, rouge, jaune, violet, vert, noir
var alphas:Array = [1, 1, 1, 1, 1, 1, 1, 1, 1];
var ratios:Array = [16, 32, 64, 96, 128, 160, 192, 224, 235];

// créer l'objet de filtre
var gradientBevel:GradientBevelFilter = new GradientBevelFilter(8, 225,
  colors, alphas, ratios, 16, 16, 1.3, 2, "inner", false);

// appliquer le filtre au clip carré
square_mc.filters = [gradientBevel];
```

Application d'un filtre de biseau dégradé

La classe GradientBevelFilter permet d'appliquer un effet de biseau dégradé à divers objets dans Flash. Un biseau dégradé est une bordure biseautée dotée d'une couleur dégradée à l'extérieur, à l'intérieur ou au-dessus d'un objet. Les bordures biseautées donnent un aspect tridimensionnel et, comme l'indique la figure suivante, peuvent produire des résultats très colorés.



Pour plus d'informations sur ce filtre, consultez la section GradientBevelFilter (flash.filters.GradientBevelFilter) dans le *Guide de référence du langage ActionScript 2.0*.

Le code suivant utilise l'API de dessin pour tracer un carré sur la scène, puis applique un filtre de biseau dégradé à la forme.

Utilisation du filtre de biseau dégradé

1. Créez un document Flash et enregistrez-le sous le nom de **gradientbevel.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
import flash.filters.GradientBevelFilter;
var shapeClip:MovieClip = this.createEmptyMovieClip("shape_mc", 1);
with (shapeClip) {
  beginFill(0xFF0000, 100);
  moveTo(0, 0);
  lineTo(200, 0);
  lineTo(200, 200);
  lineTo(0, 200);
  lineTo(0, 0);
  endFill();
}
shapeClip._x = (Stage.width - shapeClip._width) / 2;
shapeClip._y = (Stage.height - shapeClip._height) / 2;
var colors:Array = new Array(0xFFFFFF, 0xCCCCCC, 0x000000);
var alphas:Array = new Array(1, 0, 1);
var ratios:Array = new Array(0, 128, 255);
```

```
var gradientBevel:GradientBevelFilter = new GradientBevelFilter(10, 45,
  colors, alphas, ratios, 4, 4, 5, 3);
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  gradientBevel.strength++;
  shapeClip.filters = [gradientBevel];
};
mouseListener.onMouseMove = function() {
  gradientBevel.blurX = (_xmouse / Stage.width) * 255;
  gradientBevel.blurY = (_ymouse / Stage.height) * 255;
  shapeClip.filters = [gradientBevel];
};
Mouse.addListener(mouseListener);
```

Ce code utilise l'API de dessin pour tracer un carré sur la scène, placé au centre. Lorsque vous déplacez le pointeur de la souris sur la scène, le montant de flou le long des axes *x* et *y* augmente ou diminue. Le montant de flou horizontal diminue à mesure que le pointeur est déplacé vers la gauche de la scène. Lorsqu'il est déplacé vers la droite, le montant de flou augmente. Pareillement, plus le pointeur est placé haut sur la scène, plus le montant de flou le long de l'axe *y* est faible.

3. Choisissez Contrôle > Tester l'animation pour tester le document et voir les résultats.

Utilisation du filtre matrice de couleurs

La classe ColorMatrixFilter permet d'appliquer une transformation de matrice 4 x 5 aux valeurs de couleur ARVB et alpha de chaque pixel de l'image d'entrée afin d'obtenir un résultat intégrant un nouvel ensemble de valeurs de couleur ARVB et alpha. Ce filtre autorise la rotation de teinte (ombre ou couleur distincte), les modifications de saturation (intensité d'une teinte spécifique), la définition de la luminance de l'alpha (luminosité ou intensité d'une couleur) et divers autres effets. De même, vous pouvez animer ces filtres afin de créer des effets dans vos applications.

REMARQUE

Vous pouvez appliquer le filtre matrice de couleurs à des bitmaps et à des occurrences de clip.

Pour plus d'informations sur le filtre matrice de couleurs, consultez la section ColorMatrixFilter (flash.filters.ColorMatrixFilter) dans le *Guide de référence du langage ActionScript 2.0*.

Vous pouvez utiliser le filtre matrice de couleurs pour modifier la luminosité d'une occurrence, comme l'indique l'exemple suivant.

Augmentation de la luminosité d'un clip :

1. Créez un document Flash et enregistrez-le sous le nom de **brightness.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
import flash.filters.ColorMatrixFilter;
System.security.allowDomain("http://www.helpexamples.com/");
var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function(target_mc:MovieClip):Void {
  var myElements_array:Array = [1, 0, 0, 0, 100,
       0, 1, 0, 0, 100,
       0, 0, 1, 0, 100,
       0, 0, 0, 1, 0];
  var myColorMatrix_filter:ColorMatrixFilter = new
  ColorMatrixFilter(myElements_array);
  target_mc.filters = [myColorMatrix_filter];
}
this.createEmptyMovieClip("img_mc", this.getNextHighestDepth());
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcl_obj);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image2.jpg",
  img_mc);
```

   Ce code charge une image JPEG dynamiquement à l'aide d'une occurrence MovieClipLoader. Une fois l'image complètement chargée et placée sur la scène, la luminosité de l'occurrence est définie sur 100 % à l'aide d'un filtre matrice de couleurs.

3. Choisissez Contrôle > Tester l'animation pour tester le document.

Vous pouvez également créer un effet animé de luminosité en associant la classe Tween avec la classe ColorMatrixFilter comme dans l'exemple suivant.

Animation du niveau de luminosité d'une occurrence à l'aide de la classe Tween :

1. Créez un document Flash et enregistrez-le sous le nom de **brightnesstween.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
import flash.filters.ColorMatrixFilter;
import mx.transitions.Tween;
import mx.transitions.easing.*;
System.security.allowDomain("http://www.helpexamples.com");
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip):Void {
  // centrer une occurrence de clip sur la scène
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
  target_mc.watch("brightness", brightnessWatcher, target_mc);
```

```
    // animer le clip target_mc à une luminosité comprise entre
    // -100 et +100
    var myTween:Tween = new Tween(target_mc, "brightness",
    Elastic.easeOut, 100, -100, 3, true);
    myTween.onMotionFinished = function() {
      this.yoyo();
    };
};
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
    img_mc);

function brightnessWatcher(prop:String, oldVal:Number, newVal:Number,
    target_mc:MovieClip):Number {
    var brightness_array:Array = [1, 0, 0, 0, newVal,
        0, 1, 0, 0, newVal,
        0, 0, 1, 0, newVal,
        0, 0, 0, 1, 0];
    target_mc.filters = [new ColorMatrixFilter(brightness_array)];
    return newVal;
};
```

La première section du code utilise la classe MovieClipLoader pour charger une image JPEG sur la scène. Une fois l'image complètement chargée, il vous faut repositionner l'image au centre de la scène. Utilisez ensuite la classe Tween pour animer le niveau de luminosité de l'image. Pour animer la luminosité, vous utilisez la méthode `Object.watch()` qui enregistre un gestionnaire d'événements que vous démarrez dès qu'une propriété spécifiée est modifiée dans un objet ActionScript. A chaque fois qu'un code ActionScript essaie de définir la propriété de luminosité personnalisée de l'occurrence `target_mc`, vous appelez la fonction `brightnessWatcher`. La fonction personnalisée `brightnessWatcher` crée un tableau qui utilise le filtre matrice de couleurs pour définir la luminosité de l'image cible à un niveau spécifié.

3. Choisissez Contrôle > Tester l'animation pour tester le document.

   Une fois l'image chargée et placée sur la scène, la luminosité de l'image s'anime entre -100 et 100. Une fois l'interpolation de luminosité achevée, l'animation est inversée par la méthode `Tween.yoyo()` qui contraint l'interpolation à s'animer en permanence.

Utilisation du filtre convolution

La classe ConvolutionFilter applique un effet de filtre de convolution de matrice. Une convolution associe les pixels d'une image source que vous spécifiez aux pixels environnants afin de produire une image. Les convolutions permettent d'effectuer de nombreuses opérations de traitement de l'image, notamment la définition du flou, la détection de contour, l'accentuation, l'estampage et le biseautage.

REMARQUE	Vous pouvez appliquer ce filtre à des bitmaps et à des occurrences de clip.

Une convolution de matrice s'articule autour d'une matrice *n* x *m*, qui décrit la façon dont une valeur de pixels donnée dans l'image d'entrée est associée aux valeurs des pixels environnants pour obtenir une nouvelle valeur de pixels. Chaque pixel obtenu est déterminé par l'application de la matrice au pixel source correspondant et à ses pixels environnants.

Ce filtre peut uniquement être appliqué à l'aide d'ActionScript. Pour plus d'informations sur ce filtre, consultez la section ConvolutionFilter (flash.filters.ConvolutionFilter) dans le *Guide de référence du langage ActionScript 2.0*.

L'exemple suivant applique le filtre de convolution à une image JPEG chargée dynamiquement.

Utilisation du filtre de convolution pour modifier la couleur d'une image :

1. Créez un document Flash et enregistrez-le sous le nom de **convolution.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;

this.createEmptyMovieClip("shape_mc", 1);
shape_mc.createEmptyMovieClip("holder_mc", 1);
var imageLoader:MovieClipLoader = new MovieClipLoader();
imageLoader.loadClip("http://www.helpexamples.com/flash/images/
  image1.jpg", shape_mc.holder_mc);
var matrixArr:Array = [1, 4, 6, 4, 1, 4, 16, 24, 16, 4, 16, 6, 24, 36,
  24, 6, 4, 16, 24, 16, 4, 1, 4, 6, 4, 1];
var convolution:ConvolutionFilter = new ConvolutionFilter(5, 5,
  matrixArr);
shape_mc.filters = [convolution];
```

```
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  convolution.divisor = (_xmouse / Stage.width) * 271;
  convolution.bias = (_ymouse / Stage.height) * 100;
  shape_mc.filters = [convolution];
};
Mouse.addListener(mouseListener);
```

Ce code se compose de trois sections différentes. La première section importe deux classes : ConvolutionFilter et BitmapData. La deuxième section crée un clip imbriqué et utilise un objet chargeur de clip pour charger une image dans le clip imbriqué. Un objet de filtre de convolution est créé et appliqué au clip shape_mc. La dernière section de code définit un objet écouteur de souris qui modifie les propriétés de diviseur et d'écart du filtre de convolution selon la position courante du pointeur de la souris et réapplique le filtre de convolution au clip `shape_mc`.

3. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

   En déplaçant le pointeur de la souris le long de l'axe *x* de la scène, vous modifiez le diviseur du filtre, tandis que si vous le déplacez le long de l'axe *y*, vous en modifiez l'écart.

Utilisation du filtre mappage de déplacement

La classe DisplacementMapFilter utilise les valeurs de pixels de l'objet BitmapData spécifié (intitulé *image de mappage de déplacement*) pour déplacer une occurrence située sur la scène, par exemple une occurrence de clip ou de données bitmap. Vous pouvez utiliser ce filtre pour obtenir un effet voilé ou tacheté sur une occurrence spécifiée.

Ce filtre peut uniquement être appliqué à l'aide d'ActionScript. Pour plus d'informations sur ce filtre, consultez la section DisplacementMapFilter (flash.filters.DisplacementMapFilter) dans le *Guide de référence du langage ActionScript 2.0*.

Dans l'exemple suivant, une image JPEG est chargée et un filtre mappage de déplacement lui est appliqué, ce qui donne un aspect déformé à l'image. Dès que l'utilisateur déplace la souris, le mappage de déplacement est régénéré.

Déformation d'une image avec le filtre mappage de déplacement :

1. Créez un document Flash et enregistrez-le sous le nom de **displacement.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
import flash.filters.DisplacementMapFilter;
import flash.geom.Point;
import flash.display.BitmapData;

var perlinBmp:BitmapData;
var displacementMap:DisplacementMapFilter;
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip):Void  {
     target_mc._x = (Stage.width - target_mc._width) / 2;
     target_mc._y = (Stage.height - target_mc._height) / 2;
     perlinBmp = new BitmapData(target_mc._width, target_mc._height);
     perlinBmp.perlinNoise(target_mc._width, target_mc._height, 10,
  Math.round(Math.random() * 100000), false, true, 1, false);
     displacementMap = new DisplacementMapFilter(perlinBmp, new Point(0,
  0), 1, 1, 100, 100, "color");
     shapeClip.filters = [displacementMap];
};
var shapeClip:MovieClip = this.createEmptyMovieClip("shapeClip", 1);
shapeClip.createEmptyMovieClip("holderClip", 1);
var imageLoader:MovieClipLoader = new MovieClipLoader();
imageLoader.addListener(mclListener);
imageLoader.loadClip("http://www.helpexamples.com/flash/images/
  image1.jpg", shapeClip.holderClip);

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  perlinBmp.perlinNoise(shapeClip._width, shapeClip._height, 10,
  Math.round(Math.random() * 100000), false, true, 1, false);
  shapeClip.filters = [displacementMap];
};
Mouse.addListener(mouseListener);
```

Ce code charge une image JPEG et place cette image sur la scène. Une fois l'image complètement chargée, le code crée une occurrence BitmapData et utilise la méthode `perlinNoise()` pour la remplir de pixels placés aléatoirement. L'occurrence BitmapData transmet ensuite au filtre mappage de déplacement, qui est alors appliqué à l'image, la faisant apparaître déformée.

3. Choisissez Contrôle > Tester l'animation pour tester le document.

Déplacez le pointeur de la souris autour de la scène pour créer à nouveau un mappage de déplacement en appelant la méthode `perlinNoise()` qui modifie l'aspect de l'image JPEG.

Manipulation d'effets de filtre à l'aide de code

Flash permet d'ajouter dynamiquement différents filtres à vos clips, champs de texte et boutons sur la scène. Lorsque vous ajoutez et manipulez des filtres durant la lecture, vous avez la possibilité d'ajouter aussi des effets réalistes d'ombre, de flou et de rayonnement, qui réagissent aux mouvements de la souris ou aux événements utilisateur.

Pour obtenir des exemples de manipulation de filtres à l'aide de code, consultez les sections suivantes :

- « Ajustement des propriétés de filtres », à la page 526
- « Animation d'un filtre à l'aide d'ActionScript », à la page 527
- « Utilisation de la méthode clone() », à la page 528

Ajustement des propriétés de filtres

Vous pouvez accéder au tableau de filtres appliqué à un objet via des appels ActionScript normaux à l'aide de la propriété `MovieClip.filters`. L'appel de cette propriété vous renvoie un tableau contenant tous les objets de filtres présentement associés au MovieClip. Tout filtre est assorti d'un jeu de propriétés spécifique, par exemple : Les filtres sont accessibles et peuvent être modifiés comme n'importe quel objet de tableau mais, si vous utilisez la propriété `filtres`, vous obtenez un double des objets de filtre, et non pas sa référence.

La définition de la propriété `filtres` reproduit le tableau de filtres transmis et ne le stocke pas en tant que référence. Lorsque vous lisez la propriété de filtre, vous obtenez une nouvelle copie du tableau. L'un des inconvénients de cette méthodologie est que le code suivant ne fonctionne pas :

```
// ne fonctionne pas
my_mc.filters[0].blurX = 20;
```

Comme le fragment de code précédent renvoie une copie du tableau de filtres, le code modifie simplement la copie et non pas le tableau original. Pour modifier la propriété blurX, il vous faut plutôt utiliser le code ActionScript suivant :

```
// fonctionne
var filterArray:Array = my_mc.filters;
filterArray[0].blurX = 20;
my_mc.filters = filterArray;
```

L'exemple suivant applique un flou à une image à partir de la position courante du pointeur de la souris sur la scène. A chaque fois que le pointeur de la souris est déplacé horizontalement ou verticalement, les propriétés `blurX` et `blurY` du filtre de flou sont modifiées en conséquence.

Ajustement des propriétés de filtre d'un clip :

1. Créez un document Flash et enregistrez-le sous le nom de **adjustfilter.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
import flash.filters.BlurFilter;

this.createEmptyMovieClip("holder_mc", 10);
holder_mc.createEmptyMovieClip("img_mc", 20);
holder_mc.img_mc.loadMovie("http://www.helpexamples.com/flash/images/
  image2.jpg");
holder_mc.filters = [new BlurFilter(10, 10, 2)];
holder_mc._x = 75;
holder_mc._y = 75;
```

```
holder_mc.onMouseMove = function() {
  var tempFilter:BlurFilter = holder_mc.filters[0];
  tempFilter.blurX = Math.floor((_xmouse / Stage.width) * 255);
  tempFilter.blurY = Math.floor((_ymouse / Stage.height) * 255);
  holder_mc.filters = [tempFilter];
};
```

Ce code se compose de trois sections différentes. La première section importe la classe flash.filters.BlurFilter, de sorte que vous n'avez pas à utiliser le nom pleinement qualifié pour faire référence à la classe BlurFilter. La deuxième section de code crée une paire de clips et charge une image dans l'un des clips imbriqués. La troisième section répond aux mouvements de la souris sur la scène et ajuste le flou en conséquence.

3. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

   Déplacez le pointeur de la souris le long de l'axe *x*pour modifier la propriété`blurX` du filtre de flou. Déplacez le pointeur de la souris le long de l'axe *x*pour modifier la propriété`blurX` du filtre de flou. Le flou appliqué au clip diminue à mesure que le pointeur de la souris s'approche du coin supérieur gauche de la scène.

Animation d'un filtre à l'aide d'ActionScript

Vous pouvez utiliser ActionScript, par exemple la classe Tween, pour animer des filtres à l'exécution, ce qui vous permet d'appliquer des effets animés intéressants à vos applications Flash.

L'exemple suivant vous montre comment le filtre BlurFilter s'associe avec la classe Tween pour créer un flou animé qui modifie le filtre entre une valeur de 0 et une valeur de 10 à l'exécution.

Animation de flous à l'aide de la classe Tween :

1. Créez un document Flash et enregistrez-le sous le nom de **animatedfilter.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
import flash.filters.BlurFilter;
import mx.transitions.Tween;
import mx.transitions.easing.*;

this.createEmptyMovieClip("holder_mc", 10);
holder_mc.createEmptyMovieClip("img_mc", 20);
```

```
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
  var myTween:Tween = new Tween(target_mc, "blur", Strong.easeInOut, 0,
  20, 3, true);
  myTween.onMotionChanged = function() {
    target_mc._parent.filters = [new BlurFilter(target_mc.blur,
  target_mc.blur, 1)];
  };
  myTween.onMotionFinished = function() {
    myTween.yoyo();
  }
};
var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.addListener(mclListener);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  holder_mc.img_mc);
```

Ce code se compose de trois sections différentes. La première section importe les classes et les paquets requis : La deuxième section crée un clip imbriqué qui est utilisé pour charger une image et applique des filtres au clip holder. La dernière section de code crée une occurrence MovieClipLoader et un écouteur pour le chargeur de clip. L'objet écouteur définit une seule fonction de gestionnaire d'événements, `onLoadInit`, qui est lancée dès que l'image est complètement chargée et disponible sur la scène. L'image est d'abord repositionnée au centre de la scène, puis un nouvel objet Tween est créé, animant le clip et appliquant un filtre de flou de 0 et 10.

3. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

Utilisation de la méthode clone()

La méthode `clone()`, dans chaque classe de filtre, renvoie une nouvelle occurrence de filtre avec les mêmes propriétés que l'occurrence de filtre originale. Lorsque vous travaillez avec des filtres, vous pouvez avoir à effectuer une copie d'un filtre. Pour ce faire, vous devez reproduire ce filtre à l'aide de la méthode `clone()`. Si vous n'utilisez pas cette méthode pour dupliquer un filtre, Flash crée une référence au filtre original uniquement. Si Flash crée une référence au filtre original, toute modification apportée au filtre dupliqué modifie aussi l'objet de filtre original.

L'exemple suivant crée une occurrence de DropShadowFilter (`greenDropShadow`), appelle la méthode `clone()` pour reproduire le filtre d'ombre portée verte et enregistre un nouveau filtre dénommé `redDropShadow`. Le filtre cloné définit une nouvelle couleur d'ombre portée et les deux filtres sont appliqués à une occurrence de clip `flower_mc` qui se trouve sur la scène

Utilisation de la méthode clone :

1. Créez un document Flash et appelez-le **clone.fla**.
2. Créez un clip sur la scène.
3. Sélectionnez l'occurrence de clip et tapez **flower_mc** dans le champ Nom de l'occurrence de l'inspecteur des propriétés.
4. Sélectionnez l'image 1 du scénario, puis ajoutez le code suivant dans le panneau Actions :

```
import flash.filters.DropShadowFilter;
var greenDropShadow:DropShadowFilter = new DropShadowFilter();
greenDropShadow.color = 0x00FF00; // green (vert)
var redDropShadow:DropShadowFilter = greenDropShadow.clone();
redDropShadow.color = 0xFF0000; // red (rouge)
flower_mc.filters = [greenDropShadow, redDropShadow];
```

   Le code précédent crée une occurrence du filtre d'ombre portée et la nomme `greenDropShadow`. L'objet d'ombre portée verte est reproduite à l'aide de la méthode `DropShadowFilter.clone()` et crée un objet de filtre dénommé `redDropShadow`. Les deux filtres d'ombre portée verte et d'ombre portée rouge sont appliqués à l'occurrence de clip `flower_mc` sur la scène. Si vous n'avez pas appelé la méthode `clone()`, les deux ombres portées apparaissent en rouge. La raison de cette couleur rouge est que la définition de la propriété `redDropShadow.color` modifie les deux objets d'ombre portée rouge et d'ombre portée verte, car l'ombre portée rouge contient une référence à l'ombre portée verte.
5. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

   Le filtre est reproduit et cloné, puis les deux filtres sont appliqués à l'occurrence `flower_mc`.

Pour plus d'informations sur la méthode `clone()`, consultez la section clone (méthode DropShadowFilter.clone) dans le *Guide de référence du langage ActionScript 2.0*. Pour obtenir des informations connexes, vous pouvez également consulter la méthode `clone()` de n'importe quelle classe de filtre.

Création de bitmaps à l'aide de la classe BitmapData

La classe BitmapData vous permet de créer des images bitmap transparentes ou opaques aux dimensions arbitraires et de les manipuler à votre guise lors de l'exécution. En manipulant un objet BitmapData directement avec du code ActionScript, vous pouvez créer des images très complexes sans consommer de temps système supplémentaire pour le retraçage du contenu des données vectorielles dans Flash Player. Les méthodes de la classe BitmapData prennent en charge de nombreux effets qui ne sont pas disponibles via l'onglet Filtres dans l'espace de travail Flash.

Un objet BitmapData contient un tableau de données de pixels. Ces données peuvent représenter un bitmap entièrement opaque ou entièrement transparent contenant des données de canal alpha. Chaque type d'objet BitmapData est stocké en tant que tampon converti en entiers 32 bits. Chaque entier 32 bits détermine les propriétés d'un pixel unique du bitmap. Chaque entier 32 bits est une combinaison de quatre valeurs de *canal* de 8 bits (de 0 à 255) décrivant les valeurs de transparence alpha et les valeurs de rouge, vert et bleu (ARVB) du pixel.

Pour plus d'informations sur l'utilisation des paquets, consultez « Fonctionnement des paquets de filtres », à la page 493.

Pour un exemple de fichier source, BitmapData.fla, qui utilise la classe BitmapData pour manipuler une image, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/BitmapData afin d'accéder à cet exemple.

L'exemple suivant charge une image JPEG dynamiquement sur la scène et utilise la classe BitmapData pour créer un effet de bruit, analogue à un effet statique en télévision. L'effet de bruit est retracé de manière aléatoire toutes les 100 millisecondes (1/10ème de seconde). Déplacez le pointeur de la souris le long de l'axe *x* ou *y* pour déterminer la quantité du traçage statique à chaque intervalle.

Création d'un effet de bruit à l'aide de la classe BitmapData :

1. Créez un document Flash et enregistrez-le sous le nom de **noise.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
import flash.display.BitmapData;
this.createTextField("status_txt", 90, 0, 0, 100, 20);
status_txt.selectable = false;
status_txt.background = 0xFFFFFF;
status_txt.autoSize = "left";
```

```
function onMouseMove() {
  status_txt._x = _xmouse;
  status_txt._y = _ymouse-20;
  updateAfterEvent();
}
this.createEmptyMovieClip("img_mc", 10);
img_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
var noiseBmp:BitmapData = new BitmapData(Stage.width, Stage.height,
  true);
this.attachBitmap(noiseBmp, 20);
setInterval(updateNoise, 100);
var grayScale:Boolean = true;
function updateNoise():Void {
  var low:Number = 30 * _xmouse / Stage.width;
  var high:Number = 200 * _ymouse / Stage.height;
  status_txt.text = "low:" + Math.round(low) + ", high:" +
  Math.round(high);
  noiseBmp.noise(Math.round(Math.random() * 100000), low, high, 8,
  true);
}
```

Ce code crée un champ de texte avec le nom d'occurrence `status_txt`, qui suit le pointeur de la souris et affiche les valeurs courantes des paramètres `high` et `low` de la méthode `noise()`. La fonction `setInterval()` modifie l'effet de bruit, qui est mis à jour toutes les 100 millisecondes (1/10ème de seconde) en appelant la fonction `updateNoise()` sans arrêt. Les paramètres `high` et `low` de la méthode `noise()` sont déterminés en calculant la position courante du pointeur de la souris sur la scène.

3. Choisissez Contrôle > Tester l'animation pour tester le document.

   En déplaçant le pointeur de la souris le long de l'axe *x*, vous modifiez le paramètre `low` tandis que si vous le déplacez le long de l'axe *y*, vous modifiez le paramètre `high`.

La classe BitmapData permet également de déformer une image chargée dynamiquement en associant un effet de la méthode `perlinNoise()` avec un filtre de appage de déplacement. L'exemple ci-dessous l'illustre parfaitement.

Application d'un filtre mappage de déplacement à une image :

1. Créez un document Flash et enregistrez-le sous le nom de **displacement.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
// importer des classes
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
// créer un clip et un clip imbriqué
var shapeClip:MovieClip = this.createEmptyMovieClip("shapeClip", 1);
shapeClip.createEmptyMovieClip("holderClip", 1);
```

```
// charger une image JPEG
var imageLoader:MovieClipLoader = new MovieClipLoader();
imageLoader.loadClip("http://www.helpexamples.com/flash/images/
  image4.jpg", shapeClip.holderClip);
// créer une occurrence BitmapData
var perlinBmp:BitmapData = new BitmapData(Stage.width, Stage.height);
perlinBmp.perlinNoise(Stage.width, Stage.height, 10,
  Math.round(Math.random() * 100000), false, true, 1, false);
// créer et appliquer le filtre de mappage de déplacement
var displacementMap:DisplacementMapFilter = new
  DisplacementMapFilter(perlinBmp, new Point(0, 0), 1, 1, 100, 100,
  "color", 1);
shapeClip.filters = [displacementMap];
// créer et appliquer un écouteur
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  perlinBmp.perlinNoise(Stage.width, Stage.height, 10,
  Math.round(Math.random() * 100000), false, true, 1, false);
  shapeClip.filters = [displacementMap];
}
Mouse.addListener(mouseListener);
```

Ce code comporte cinq sections logiques. La première section importe les classes nécessaires pour l'exemple. La deuxième section de code crée un clip imbriqué et charge une image JPEG depuis un serveur distant. Le troisième bloc de code crée une occurrence BitmapData dénommée `perlinBmp` aux mêmes dimensions que la scène. L'occurrence `perlinBmp` contient les résultats d'un effet de bruit Perlin et elle est utilisée par la suite comme paramètre pour le filtre de mappage de déplacement. Le quatrième bloc crée et applique un effet de filtre mappage de déplacement à l'image chargée dynamiquement créée précédemment. Le cinquième et dernier bloc crée un écouteur pour la souris, qui régénère le bruit Perlin utilisé par le filtre mappage de déplacement à chaque fois que l'utilisateur déplace son pointeur.

3. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

A propos des modes de fondu

Vous pouvez appliquer des modes de fondu à des objets de clip via l'espace de travail Flash ou ActionScript. A l'exécution, plusieurs graphiques sont fusionnés en une seule forme. C'est pourquoi vous ne pouvez pas appliquer différents modes de fondu à différents symboles graphiques.

Pour plus d'informations sur l'application des modes de fondu à l'aide d'ActionScript, consultez « Application de modes de fondu », à la page 533.

Les modes de fondu supposent d'associer les couleurs d'une image (l'image de base) à celles d'une autre image (l'image fondue) afin de produire une troisième image. Chaque valeur de pixels d'une image est traitée avec la valeur de pixels correspondante de l'autre image afin d'obtenir un résultat présentant une valeur de pixels de position identique.

La propriété `MovieClip.blendMode` prend en charge les modes de fondu suivants :

add S'utilise couramment pour créer un effet de dissolution animée entre deux images en éclaircissant progressivement leurs couleurs.

alpha S'utilise couramment pour appliquer la transparence de l'avant-plan à l'arrière-plan.

darken S'utilise couramment pour superposer un type.

difference S'utilise couramment pour créer des couleurs plus vives.

erase S'utilise couramment pour effacer (*cut out*) une partie de l'arrière-plan à l'aide de l'alpha d'avant-plan.

hardlight S'utilise couramment pour créer des effets d'ombrage.

invert S'utilise pour inverser l'arrière-plan.

layer S'utilise pour forcer la création d'un tampon provisoire en vue de la précomposition d'un clip spécifique.

lighten S'utilise couramment pour superposer un type.

multiply S'utilise couramment pour créer des ombres et des effets de profondeur.

normal S'utilise pour que les valeurs des pixels de l'image fondue supplantent celles de l'image de base.

overlay S'utilise couramment pour créer des effets d'ombrage.

screen S'utilise couramment pour créer des surlignements et des halos.

subtract S'utilise couramment pour créer un effet de dissolution animée en assombrissant progressivement leurs couleurs.

Application de modes de fondu

L'exemple suivant charge une image dynamique et permet d'appliquer différents modes de fondu à l'image en sélectionnant un mode à partir d'un composant ComboBox sur la scène.

Application de différents modes de fondu à une image :

1. Créez un document Flash et enregistrez-le sous le nom de **blendmodes.fla**.
2. Faites glisser une occurrence de composant ComboBox sur la scène et attribuez-lui le nom de **blendMode_cb**.

3. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
var blendMode_dp:Array = new Array();
blendMode_dp.push({data:"add", label:"add"});
blendMode_dp.push({data:"alpha", label:"alpha"});
blendMode_dp.push({data:"darken", label:"darken"});
blendMode_dp.push({data:"difference", label:"difference"});
blendMode_dp.push({data:"erase", label:"erase"});
blendMode_dp.push({data:"hardlight", label:"hardlight"});
blendMode_dp.push({data:"invert", label:"invert"});
blendMode_dp.push({data:"layer", label:"layer"});
blendMode_dp.push({data:"lighten", label:"lighten"});
blendMode_dp.push({data:"multiply", label:"multiply"});
blendMode_dp.push({data:"normal", label:"normal"});
blendMode_dp.push({data:"overlay", label:"overlay"});
blendMode_dp.push({data:"screen", label:"screen"});
blendMode_dp.push({data:"subtract", label:"subtract"});
blendMode_cb.dataProvider = blendMode_dp;

var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  var blendModeClip:MovieClip =
  target_mc.createEmptyMovieClip("blendModeType_mc", 20);
  with (blendModeClip) {
    beginFill(0x999999);
    moveTo(0, 0);
    lineTo(target_mc._width / 2, 0);
    lineTo(target_mc._width / 2, target_mc._height);
    lineTo(0, target_mc._height);
    lineTo(0, 0);
    endFill();
  }
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
  blendModeClip.blendMode = blendMode_cb.value;
};
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);

function cbListener(eventObj:Object):Void {
  img_mc.blendModeType_mc.blendMode = eventObj.target.value;
}
blendMode_cb.addEventListener("change", cbListener);
```

Ce code ActionScript remplit le composant ComboBox de chaque type de mode de fondu, de manière à ce que l'utilisateur puisse visualiser chaque effet sur l'image chargée dynamiquement. Un objet écouteur est créé afin d'être utilisé avec une occurrence MovieClipLoader. L'objet écouteur définit un seul écouteur d'événements, `onLoadInit`, qui est lancé dès que l'image est chargée complètement et initialisée par Flash. L'écouteur d'événements crée un clip dénommé `blendModeType_mc` et utilise l'API de dessin pour tracer une forme rectangulaire sur la moitié gauche de l'image. Le mode de fondu sélectionné présentement pour l'occurrence ComboBox est ensuite appliqué au clip `blendModeType_mc`.

Le reste du code configure l'occurrence MovieClipLoader, qui charge l'image spécifiée dans un clip sur la scène. Enfin, un écouteur est défini pour l'occurrence ComboBox `blendMode_cb` qui applique le mode de fondu sélectionné dès qu'un nouvel élément est sélectionné à partir de l'occurrence ComboBox.

4. Sélectionnez Contrôle > Tester l'animation pour tester le document.

Présentation de l'ordre des opérations

La liste suivante représente l'ordre des opérations dans lequel un tableau de filtres, des modes de fondu, des transformations de couleurs et des calques de masque sont associés ou exécutés pour une occurrence de clip :

1. Le bitmap du clip est mis à jour à partir du contenu vectoriel (la propriété `cacheAsBitmap` est définie sur `true`).
2. Si vous utilisez la méthode `setMask()` et si le masque a un cache de bitmaps, Flash effectue un fondu alpha entre les deux images.
3. Des filtres sont alors appliqués (flou, ombre portée, rayonnement, etc.).
4. Si vous utilisez la classe ColorTransform, l'opération de transformation des couleurs est effectuée et mise en cache de bitmaps.
5. Si vous appliquez un mode de fondu, le fondu est alors exécuté (avec un rendu vectoriel).
6. Si vous appliquez des calques de masque externes, les calques exécutent le masquage (avec un rendu vectoriel).

Dessin à l'aide du code ActionScript

Vous pouvez utiliser des méthodes de la classe MovieClip pour dessiner des lignes et des remplissages sur la scène. Vous pouvez ainsi créer des outils de dessin pour les utilisateurs et tracer des formes dans le fichier SWF en réponse à des événements. Les méthodes de dessin de la classe MovieClip sont les suivantes :

- `beginFill()`
- `beginGradientFill()`
- `clear()`
- `curveTo()`
- `endFill()`
- `lineTo()`
- `lineStyle()`
- `moveTo()`

Vous pouvez utiliser les méthodes de dessin avec n'importe quel clip. Toutefois, si vous les utilisez dans un clip créé en mode auteur, les méthodes de dessin sont exécutées avant que le clip ne soit dessiné. En d'autres termes, le contenu créé en mode auteur est dessiné par-dessus le contenu dessiné à l'aide des méthodes de dessin.

Vous pouvez utiliser des clips avec des méthodes de dessin en tant que masques. Toutefois, comme pour tous les masques de clips, les traits seront ignorés.

Pour plus d'informations sur le dessin à l'aide de code ActionScript, consultez les sections suivantes :

- « Utilisation des méthodes de dessin pour tracer des lignes, des courbes et des formes », à la page 537
- « Dessin de formes spéciales », à la page 539
- « Utilisation de remplissages dégradés complexes », à la page 542
- « Utilisation de styles de ligne », à la page 543
- « Utilisation de méthodes API et mise en script d'animation », à la page 550

Pour un exemple de fichier source, drawingapi.fla, qui vous décrit comment utiliser l'application API de dessin, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/DrawingAPI afin d'accéder à cet exemple.

Utilisation des méthodes de dessin pour tracer des lignes, des courbes et des formes

Vous pouvez utiliser l'API de dessin de Flash pour créer des formes dynamiquement sur la scène à l'exécution. Vous pouvez utiliser ces formes pour masquer des contenus dynamiquement, leur appliquer des filtres ou les animer sur la scène. Vous pouvez également utiliser l'API de dessin pour créer différents outils de dessin, permettant aux utilisateurs de tracer des formes au moyen de leur souris ou de leur clavier dans le fichier SWF.

Dessin d'une ligne :

1. Créez un document Flash et enregistrez-le sous le nom de **line.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
this.createEmptyMovieClip("line_mc", 10);
line_mc.lineStyle(1, 0x000000, 100);
line_mc.moveTo(0, 0);
line_mc.lineTo(200, 100);
line_mc._x = 100;
line_mc._y = 100;
```

   Ce code trace sur la scène une ligne de 0,0 à 200,100. Les coordonnées `_x` et `_y` de la ligne sont ensuite modifiées de manière à repositionner la ligne à 100,100 sur la scène.
3. Enregistrez le document Flash, puis choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

Pour dessiner une forme plus complexe, continuez d'appeler la méthode `MovieClip.lineTo()` et dessinez un rectangle, un carré ou un ovale, comme l'indiquent les exemples ci-dessous.

Dessin d'une courbe :

1. Créez un document Flash et enregistrez-le sous le nom de **curve.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
this.createEmptyMovieClip("circle_mc", 1);
with (circle_mc) {
  lineStyle(4, 0x000000, 100);
  beginFill(0xFF0000);
  moveTo(200, 300);
  curveTo(300, 300, 300, 200);
  curveTo(300, 100, 200, 100);
  curveTo(100, 100, 100, 200);
  curveTo(100, 300, 200, 300);
  endFill();
}
```

3. Enregistrez le document Flash, puis choisissez Contrôle > Tester l'animation pour tester le fichier.

   Ce code utilise l'API de dessin pour tracer un cercle sur la scène. La forme du cercle utilise seulement quatre appels à la méthode `MovieClip.curveTo()` et peut donc apparaître légèement déformée. Pour obtenir un autre exemple de dessin de cercle à l'aide de l'API de dessin, consultez l'exemple du cercle décrit à la section « Dessin de formes spéciales », à la page 539 ; le code effectue huit appels à la méthode `MovieClip.curveTo()` afin de dessiner un cercle plus réaliste.

Dessin d'un triangle :

1. Créez un document Flash et enregistrez-le sous le nom de **triangle.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

   ```
   this.createEmptyMovieClip("triangle_mc", 1);
   ```

   Ce code utilise la méthode `MovieClip.createEmptyMovieClip()` pour créer un clip vide sur la scène. Le nouveau clip est l'enfant d'un clip existant (dans le cas présent, le scénario principal).
3. Ajoutez le code ActionScript suivant à l'image 1 du scénario, à la suite du code ajouté à l'étape précédente :

   ```
   with (triangle_mc) {
     lineStyle(5, 0xFF00FF, 100);
     moveTo(200, 200);
     lineTo(300, 300);
     lineTo(100, 300);
     lineTo(200, 200);
   }
   ```

   Dans ce code, le clip vide (`triangle_mc`) appelle des méthodes de dessin. Ce code trace un triangle avec des lignes violettes de 5 pixels, sans remplissage.
4. Enregistrez le document Flash, puis choisissez Contrôle > Tester l'animation pour tester le document Flash.

Pour plus d'informations sur ces méthodes, consultez leurs rubriques respectives dans `MovieClip` dans le *Guide de référence du langage ActionScript 2.0*.

Pour un exemple de fichier source, drawingapi.fla, qui vous décrit comment utiliser l'API de dessin dans une application Flash, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/DrawingAPI afin d'accéder à cet exemple.

Dessin de formes spéciales

Cette section explique comment créer des méthodes plus flexibles à utiliser pour dessiner des formes plus complexes, par exemple des rectangles arrondis et des cercles.

Création d'un rectangle :

1. Créez un document Flash et enregistrez-le sous le nom de **rect.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
this.createEmptyMovieClip("rectangle_mc", 10);
rectangle_mc._x = 100;
rectangle_mc._y = 100;
drawRectangle(rectangle_mc, 240, 180, 0x99FF00, 100);
function drawRectangle(target_mc:MovieClip, boxWidth:Number,
  boxHeight:Number, fillColor:Number, fillAlpha:Number):Void {
  with (target_mc) {
    beginFill(fillColor, fillAlpha);
    moveTo(0, 0);
    lineTo(boxWidth, 0);
    lineTo(boxWidth, boxHeight);
    lineTo(0, boxHeight);
    lineTo(0, 0);
    endFill();
  }
}
```

3. Enregistrez le document Flash, puis choisissez Contrôle > Tester l'animation pour tester le fichier.

   Flash dessine un simple rectangle vert sur la scène et le positionne à 100,100.
   Pour modifier les dimensions du rectangle, sa couleur de remplissage ou sa transparence, vous pouvez modifier les valeurs correspondantes dans l'appel à la méthode `drawRectangle()`, sans avoir à modifier le contenu de la méthode `MovieClip.beginFill()`.

Vous pouvez aussi créer un rectangle aux coins arrondis à l'aide de l'API de dessin, comme le montre l'exemple suivant.

Création d'un rectangle arrondi :

1. Créez un document Flash et enregistrez-le sous le nom de **roundrect.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
this.createEmptyMovieClip("rectangle_mc", 10);
rectangle_mc._x = 100;
rectangle_mc._y = 100;
drawRoundedRectangle(rectangle_mc, 240, 180, 20, 0x99FF00, 100);
function drawRoundedRectangle(target_mc:MovieClip, boxWidth:Number,
  boxHeight:Number, cornerRadius:Number, fillColor:Number,
  fillAlpha:Number):Void {
  with (target_mc) {
    beginFill(fillColor, fillAlpha);
    moveTo(cornerRadius, 0);
    lineTo(boxWidth - cornerRadius, 0);
    curveTo(boxWidth, 0, boxWidth, cornerRadius);
    lineTo(boxWidth, cornerRadius);
    lineTo(boxWidth, boxHeight - cornerRadius);
    curveTo(boxWidth, boxHeight, boxWidth - cornerRadius, boxHeight);
    lineTo(boxWidth - cornerRadius, boxHeight);
    lineTo(cornerRadius, boxHeight);
    curveTo(0, boxHeight, 0, boxHeight - cornerRadius);
    lineTo(0, boxHeight - cornerRadius);
    lineTo(0, cornerRadius);
    curveTo(0, 0, cornerRadius, 0);
    lineTo(cornerRadius, 0);
    endFill();
  }
}
```

3. Enregistrez le document Flash, puis choisissez Contrôle > Tester l'animation pour tester le fichier.

   Un rectangle vert apparaît sur la scène, d'une largeur de 240 pixels, d'une hauteur de 180 pixels et doté de coins arrondis de 20 pixels. Vous pouvez créer plusieurs occurrences de rectangles arrondis en créant des clips à l'aide de `MovieClip.createEmptyMovieClip()` et en appelant votre fonction `drawRoundedRectangle()` personnalisée.

Vous pouvez créer un cercle parfait à l'aide de l'API de dessin comme le montre l'exemple suivant.

Création d'un cercle :

1. Créez un document Flash et enregistrez-le sous le nom de **circle2.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
this.createEmptyMovieClip("circle_mc", 10);
circle_mc._x = 100;
circle_mc._y = 100;
drawCircle(circle_mc, 100, 0x99FF00, 100);

function drawCircle(target_mc:MovieClip, radius:Number,
  fillColor:Number, fillAlpha:Number):Void {
  var x:Number = radius;
  var y:Number = radius;
  with (target_mc) {
    beginFill(fillColor, fillAlpha);
    moveTo(x + radius, y);
    curveTo(radius + x, Math.tan(Math.PI / 8) * radius + y,
  Math.sin(Math.PI / 4) * radius + x, Math.sin(Math.PI / 4) * radius +
  y);
    curveTo(Math.tan(Math.PI / 8) * radius + x, radius + y, x, radius +
  y);
    curveTo(-Math.tan(Math.PI / 8) * radius + x, radius+ y, -
  Math.sin(Math.PI / 4) * radius + x, Math.sin(Math.PI / 4) * radius +
  y);
    curveTo(-radius + x, Math.tan(Math.PI / 8) * radius + y, -radius +
  x, y);
    curveTo(-radius + x, -Math.tan(Math.PI / 8) * radius + y, -
  Math.sin(Math.PI / 4) * radius + x, -Math.sin(Math.PI / 4) * radius +
  y);
    curveTo(-Math.tan(Math.PI / 8) * radius + x, -radius + y, x, -radius
  + y);
    curveTo(Math.tan(Math.PI / 8) * radius + x, -radius + y,
  Math.sin(Math.PI / 4) * radius + x, -Math.sin(Math.PI / 4) * radius +
  y);
    curveTo(radius + x, -Math.tan(Math.PI / 8) * radius + y, radius + x,
  y);
    endFill();
  }
}
```

3. Enregistrez le document Flash, puis choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

Ce code crée un cercle plus complexe, et plus réaliste, que l'exemple de cercle précédent. Au lieu des quatre appels à la méthode `curveTo()`, cet exemple utilise huit appels à la méthode `curveTo()`, ce qui donne une forme d'aspect bien plus arrondi.

Vous pouvez utiliser l'API de dessin pour créer un triangle comme le montre l'exemple suivant.

Création d'un triangle de fantaisie

1. Créez un document Flash et enregistrez-le sous le nom de **fancytriangle.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
this.createEmptyMovieClip("triangle_mc", 10);
triangle_mc._x = 100;
triangle_mc._y = 100;
drawTriangle(triangle_mc, 100, 0x99FF00, 100);

function drawTriangle(target_mc:MovieClip, sideLength:Number,
  fillColor:Number, fillAlpha:Number):Void {
  var tHeight:Number = sideLength * Math.sqrt(3) / 2;
  with (target_mc) {
    beginFill(fillColor, fillAlpha);
    moveTo(sideLength / 2, 0);
    lineTo(sideLength, tHeight);
    lineTo(0, tHeight);
    lineTo(sideLength / 2, 0);
    endFill();
  }
}
```

   L'API de dessin trace un triangle équilatéral sur la scène et le remplit d'une couleur et d'un montant alpha (transparence) spécifiés.
3. Enregistrez le document Flash, puis choisissez Contrôle > Tester l'animation pour tester le fichier.

Pour un exemple de fichier source, drawingapi.fla, qui vous décrit comment utiliser l'API de dessin dans une application Flash, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/DrawingAPI afin d'accéder à cet exemple.

Utilisation de remplissages dégradés complexes

L'API de dessin de Flash prend en charge les remplissages dégradés, ainsi que les remplissages unis. L'exemple suivant crée un clip sur la scène, utilise l'API de dessin pour créer un carré, puis remplit ce carré avec un dégradé radial rouge et bleu.

Création d'un dégradé complexe :

1. Créez un document Flash et enregistrez-le sous le nom de **radialgradient.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
this.createEmptyMovieClip("gradient_mc", 10);
var fillType:String = "radial";
var colors:Array = [0xFF0000, 0x0000FF];
var alphas:Array = [100, 100];
var ratios:Array = [0, 0xFF];
var matrix:Object = {a:200, b:0, c:0, d:0, e:200, f:0, g:200, h:200,
  i:1};
var spreadMethod:String = "reflect";
var interpolationMethod:String = "linearRGB";
var focalPointRatio:Number = 0.9;
with (gradient_mc) {
  beginGradientFill(fillType, colors, alphas, ratios, matrix,
  spreadMethod, interpolationMethod, focalPointRatio);
  moveTo(100, 100);
  lineTo(100, 300);
  lineTo(300, 300);
  lineTo(300, 100);
  lineTo(100, 100);
  endFill();
}
```

   Le code ActionScript précédent utilise l'API de dessin pour créer un carré sur la scène et appelle la méthode beginGradientFill() pour remplir ce carré d'un dégradé circulaire rouge et bleu.
3. Enregistrez le document Flash, puis choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

Utilisation de styles de ligne

L'API de dessin de Flash vous permet de spécifier un style de ligne que Flash utilisera ensuite pour ses appels à `MovieClip.lineTo()` et `MovieClip.curveTo()` tant que vous n'aurez pas appelé `MovieClip.lineStyle()` avec des paramètres différents comme l'indique l'exemple suivant :

```
lineStyle(thickness:Number, rgb:Number, alpha:Number, pixelHinting:Boolean,
  noScale:String, capsStyle:String, jointStyle:String, miterLimit:Number)
```

Vous pouvez appeler `MovieClip.lineStyle()` au cours du dessin afin de spécifier différents styles pour divers segments de ligne dans un tracé.

Pour plus d'informations sur la définition de styles de ligne à l'aide du code ActionScript, consultez les sections suivantes :

- « Définition de styles de trait et d'extrémités », à la page 544
- « Définition des paramètres des styles de ligne », à la page 545

Définition de styles de trait et d'extrémités

Flash 8 comporte diverses améliorations au dessin de lignes. Les nouveaux paramètres de ligne ajoutés à Flash Player 8 sont `pixelHinting`, `noScale`, `capsStyle`, `jointStyle` et `miterLimit`.

L'exemple suivant explique la différence entre les trois nouveaux styles d'extrémités dans Flash Player 8 : `none`, `round` et `square`.

Définition de styles d'extrémités à l'aide du code ActionScript :

1. Créez un document Flash et enregistrez-le sous le nom de **capstyle.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
// configurer une grille de clip
this.createEmptyMovieClip("grid_mc", 50);
grid_mc.lineStyle(0, 0x999999, 100);
grid_mc.moveTo(50, 0);
grid_mc.lineTo(50, Stage.height);
grid_mc.moveTo(250, 0);
grid_mc.lineTo(250, Stage.height);
// ligne 1 (capsStyle : arrondi)
this.createEmptyMovieClip("line1_mc", 10);
with (line1_mc) {
  createTextField("label_txt", 1, 5, 10, 100, 20);
  label_txt.text = "round";
  lineStyle(20, 0x99FF00, 100, true, "none", "round", "miter", 0.8);
  moveTo(0, 0);
  lineTo(200, 0);
  _x = 50;
  _y = 50;
}
// ligne 2 (capsStyle : carré)
this.createEmptyMovieClip("line2_mc", 20);
with (line2_mc) {
  createTextField("label_txt", 1, 5, 10, 100, 20);
  label_txt.text = "square";
  lineStyle(20, 0x99FF00, 100, true, "none", "square", "miter", 0.8);
  moveTo(0, 0);
  lineTo(200, 0);
  _x = 50;
  _y = 150;
}
// ligne 3 (capsStyle : aucun)
this.createEmptyMovieClip("line3_mc", 30);
with (line3_mc) {
  createTextField("label_txt", 1, 5, 10, 100, 20);
  label_txt.text = "none";
  lineStyle(20, 0x99FF00, 100, true, "none", "none", "miter", 0.8);
  moveTo(0, 0);
  lineTo(200, 0);
  _x = 50;
  _y = 250;
}
```

Le code précédent crée quatre clips dynamiquement et utilise l'API de dessin pour créer une série de lignes sur la scène. Le premier clip contient deux lignes verticales, l'une de 50 pixels et l'autre de 250 pixels sur l'axe x. Les trois autres clips tracent chacun une ligne verte sur la scène et définissent leur CapsStyle sur arrondi, carré ou aucun.

3. Choisissez Contrôle > Tester l'animation pour tester le document.

   Les différents styles d'extrémités apparaissent sur la scène à l'exécution.

Définition des paramètres des styles de ligne

Vous pouvez définir les paramètres des styles de ligne afin de modifier l'aspect de vos traits. Vous pouvez ainsi utiliser des paramètres pour modifier l'épaisseur, la couleur, l'alpha, l'échelle et d'autres attributs du style de ligne.

Définition de l'épaisseur de ligne

Le paramètre `thickness` de la méthode `MovieClip.lineStyle()` permet de spécifier l'épaisseur de la ligne tracée en points sous la forme d'un nombre. Vous pouvez tracer une ligne de n'importe quelle épaisseur comprise entre 0 et 255 point(s). Toutefois, si vous définissez l'épaisseur sur 0, vous créez ce que l'on appelle un *filet*, c'est-à-dire que le trait sera toujours de 1 pixel, même si un zoom ou un redimensionnement est pratiqué sur le fichier SWF.

L'exemple suivant explique la différence entre une ligne standard de 1 pixel d'épaisseur et un filet.

Création d'un filet :

1. Créez un document Flash et enregistrez-le sous le nom de **hairline.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
this.createEmptyMovieClip("drawing_mc", 10);
// création d'un filet rouge
drawing_mc.lineStyle(0, 0xFF0000, 100);
drawing_mc.moveTo(0, 0);
drawing_mc.lineTo(200, 0);
drawing_mc.lineTo(200, 100);
// création d'une ligne bleue d'une épaisseur de 1 pixel
drawing_mc.lineStyle(1, 0x0000FF, 100);
drawing_mc.lineTo(0, 100);
drawing_mc.lineTo(0, 0);
drawing_mc._x = 100;
drawing_mc._y = 100;
```

   Le code précédent utilise l'API de dessin pour tracer deux lignes sur la scène. La première ligne est rouge et présente une épaisseur de 0, ce qui indique qu'il s'agit d'un filet. La seconde ligne est bleue et présente une épaisseur de 1 pixel.

3. Enregistrez le document Flash, puis choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

   Au départ, la ligne rouge et la ligne bleue semblent parfaitement identiques. Toutefois, si vous cliquez du bouton droit sur le fichier SWF et choissez Zoom In dans le menu contextuel, la ligne rouge apparaît sous la forme d'une ligne de 1 pixel en revanche, la ligne bleue s'agrandit à chaque fois que vous zoomez dans le fichier SWF.

Définition d'une couleur de ligne (RVB)

Le deuxième paramètre dans la méthode `lineStyle()`, `RVB`, vous permet de contrôler la couleur du segment courant de ligne sous la forme d'un nombre. Par défaut, Flash trace des lignes noires (`#000000`), mais vous pouvez spécifier des couleurs différentes en définissant une nouvelle valeur de couleur en hexadécimal à l'aide de la syntaxe `0xRRGGBB`. Dans cette syntaxe, `RR` désigne une valeur rouge (entre `00` et `FF`), `GG` une valeur verte (`00` à `FF`) et `BB` une valeur bleue (`00` à `FF`).

Par exemple, vous représentez une ligne rouge par le nombre `0xFF0000`, une ligne verte par le nombre `0x00FF00`, une ligne bleue par `0x0000FF`, une ligne violette par `0xFF00FF` (rouge et bleue), une ligne blanche par `#FFFFFF`, une ligne grise par `#999999`, et ainsi de suite.

Définition d'un alpha de ligne

Le troisième paramètre dans la méthode `lineStyle()`, `alpha`, vous permet de contrôler le niveau de transparence (alpha) d'une ligne. La transparence est une valeur numérique comprise entre 0 et 100, où 0 représente une ligne totalement transparente et 100 une ligne totalement opaque (visible).

Définition du lissage des pixels d'une ligne (pixelHinting)

Le lissage des pixels pour le paramètre des traits, `pixelHinting`, signifie que les points d'ancrage d'une ligne et d'une courbe sont définis sur des pixels entiers. Les traits sont sur des pixels entiers pour toute épaisseur de trait, ce qui signifie que vous ne verrez jamais de ligne verticale ou horizontale floue. Vous définissez le paramètre `pixelHinting` sur une valeur booléenne (`true` ou `false`).

Définition du redimensionnement d'une ligne (noScale)

Vous définissez le paramètre`noScale` à l'aide d'une valeur String, ce qui vous permet de spécifier un mode de redimensionnement pour la ligne. Vous pouvez utiliser un trait non redimensionnable en mode horizontal ou en mode vertical, redimensionner la ligne (normal) ou ne pas utiliser le redimensionnement.

CONSEIL	Il est utile d'autoriser le redimensionnement pour les éléments de l'interface utilisateur lorsque les utilisateurs effectuent un zoom. C'est en revanche inutile si le clip est seulement redimensionné à la verticale ou à l'horizontale.

Vous disposez de quatre modes différents pour spécifier à quel moment un redimensionnement doit être pratiqué et quand il doit être évité. Les valeurs suivantes sont ainsi possibles pour la propriété `noScale` :

`normal` Ajuste toujours l'épaisseur (*par défaut*).

`vertical` N'ajuste pas l'épaisseur si l'objet est redimensionné verticalement.

`horizontal` N'ajuste pas l'épaisseur si l'objet est redimensionné horizontalement.

`none` N'ajuste jamais l'épaisseur.

Définition d'extrémités (capsStyle) et de liaisons (jointStyle) de ligne

Vous pouvez définir trois types de styles d'extrémités pour le paramètre `capsStyle` :

- `round` (*par défaut*)
- `square`
- `none`

L'exemple suivant explique les différences entre chacun des trois styles d'extrémités. Une représentation visuelle de chaque style d'extrémité apparaît sur la scène lorsque vous testez le fichier SWF.

Définition de différents styles d'extrémités :

1. Créez un document Flash et enregistrez-le sous le nom de **capstyle2.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
var lineLength:Number = 100;
// arrondi
this.createEmptyMovieClip("round_mc", 10);
round_mc.lineStyle(20, 0xFF0000, 100, true, "none", "round");
round_mc.moveTo(0, 0);
round_mc.lineTo(lineLength, 0);
round_mc.lineStyle(0, 0x000000);
round_mc.moveTo(0, 0);
round_mc.lineTo(lineLength, 0);
round_mc._x = 50;
round_mc._y = 50;
var lbl:TextField = round_mc.createTextField("label_txt", 10, 0, 10,
  lineLength, 20);
lbl.text = "round";
var lineLength:Number = 100;
// carré
this.createEmptyMovieClip("square_mc", 20);
square_mc.lineStyle(20, 0xFF0000, 100, true, "none", "square");
square_mc.moveTo(0, 0);
square_mc.lineTo(lineLength, 0);
square_mc.lineStyle(0, 0x000000);
square_mc.moveTo(0, 0);
```

```
square_mc.lineTo(lineLength, 0);
square_mc._x = 200;
square_mc._y = 50;
var lbl:TextField = square_mc.createTextField("label_txt", 10, 0, 10,
  lineLength, 20);
lbl.text = "square";
// aucun
this.createEmptyMovieClip("none_mc", 30);
none_mc.lineStyle(20, 0xFF0000, 100, true, "none", "none");
none_mc.moveTo(0, 0);
none_mc.lineTo(lineLength, 0);
none_mc.lineStyle(0, 0x000000);
none_mc.moveTo(0, 0);
none_mc.lineTo(lineLength, 0);
none_mc._x = 350;
none_mc._y = 50;
var lbl:TextField = none_mc.createTextField("label_txt", 10, 0, 10,
  lineLength, 20);
lbl.text = "none";
```

Le code précédent utilise l'API de dessin pour tracer trois lignes, chacune dotée d'un style d'extrémité distinct.

3. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

Vous pouvez définir les trois types suivants de styles de liaison pour le paramètre `jointStyle` :

- `round` (*par défaut*)
- `miter`
- `bevel`

L'exemple suivant explique les différences entre chacun des trois styles de liaison :

Définition de différents styles de liaison :

1. Créez un document Flash et enregistrez-le sous le nom de **jointstyles.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
var lineLength:Number = 100;
// en pointe
this.createEmptyMovieClip("miter_mc", 10);
miter_mc.lineStyle(25, 0xFF0000, 100, true, "none", "none", "miter",
  25);
miter_mc.moveTo(0, lineLength);
miter_mc.lineTo(lineLength / 2, 0);
miter_mc.lineTo(lineLength, lineLength);
miter_mc.lineTo(0, lineLength);
miter_mc._x = 50;
miter_mc._y = 50;
var lbl:TextField = miter_mc.createTextField("label_txt", 10, 0,
  lineLength + 20, lineLength, 20);
```

```
lbl.autoSize = "center";
lbl.text = "miter";
// arrondi
this.createEmptyMovieClip("round_mc", 20);
round_mc.lineStyle(25, 0xFF0000, 100, true, "none", "none", "round");
round_mc.moveTo(0, lineLength);
round_mc.lineTo(lineLength / 2, 0);
round_mc.lineTo(lineLength, lineLength);
round_mc.lineTo(0, lineLength);
round_mc._x = 200;
round_mc._y = 50;
var lbl:TextField = round_mc.createTextField("label_txt", 10, 0,
  lineLength + 20, lineLength, 20);
lbl.autoSize = "center";
lbl.text = "round";
// biseau
this.createEmptyMovieClip("bevel_mc", 30);
bevel_mc.lineStyle(25, 0xFF0000, 100, true, "none", "none", "bevel");
bevel_mc.moveTo(0, lineLength);
bevel_mc.lineTo(lineLength / 2, 0);
bevel_mc.lineTo(lineLength, lineLength);
bevel_mc.lineTo(0, lineLength);
bevel_mc._x = 350;
bevel_mc._y = 50;
var lbl:TextField = bevel_mc.createTextField("label_txt", 10, 0,
  lineLength + 20, lineLength, 20);
lbl.autoSize = "center";
lbl.text = "bevel";
```

Flash utilise l'API de dessin pour tracer trois triangles sur la scène. Chaque triangle a une valeur différente pour son style de liaison.

3. Enregistrez le document Flash, puis choisissez Contrôle > Tester l'animation pour tester le fichier.

Définition d'une pointe de ligne (miterLimit)

La propriété `miterLimit` est une valeur numérique qui indique la limite à laquelle une liaison de pointes (voir « Définition d'extrémités (capsStyle) et de liaisons (jointStyle) de ligne », à la page 547) est coupée. La valeur `miterLimit` est un multiplicateur général de trait. Par exemple, avec une valeur de 2,5, `miterLimit` est coupé à 2,5 fois la taille du trait. La plage de valeurs valides est de 0 à 255 (si une valeur `miterLimit` n'est pas définie, la valeur par défaut est `3`). La propriété `miterLimit` n'est utilisée que si `jointStyle` est défini sur `pointu`.

Utilisation de méthodes API et mise en script d'animation

Vous pouvez associer l'API de dessin avec les classes Tween et TransitionManager pour créer d'excellents résultats animés, auquel cas vous n'avez que très peu de code ActionScript à rédiger.

L'exemple suivant charge une image JPEG et la masque dynamiquement de manière à pouvoir révéler lentement l'image chargée par interpolation du masque.

Animation de masques dynamiques :

1. Créez un document Flash et enregistrez-le sous le nom de **dynmask.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario :

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  target_mc._visible = false;
  // centrer l'image sur la scène
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
  var maskClip:MovieClip = target_mc.createEmptyMovieClip("mask_mc",
  20);
  with (maskClip) {
    // dessiner un masque de mêmes dimensions que l'image chargée
    beginFill(0xFF00FF, 100);
    moveTo(0, 0);
    lineTo(target_mc._width, 0);
    lineTo(target_mc._width, target_mc._height);
    lineTo(0, target_mc._height);
    lineTo(0, 0);
    endFill();
  }
  target_mc.setMask(maskClip);
  target_mc._visible = true;
  var mask_tween:Tween = new Tween(maskClip, "_yscale", Strong.easeOut,
  0, 100, 2, true);
};
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);
```

Cet exemple de code importe la classe Tween ainsi que chacune des classes du paquet d'accélération. Ensuite, il crée un objet faisant fonction d'objet écouteur pour une occurrence MovieClipLoader, créée par une autre section du code. L'objet écouteur définit un seul gestionnaire d'événements, `onLoadInit` qui centre sur la scène l'image JPEG chargée dynamiquement. Une fois que le code a repositionné l'image, une occurrence de clip est créée dans le clip `target_mc` (contenant l'image JPEG chargée dynamiquement). Le code de l'API de dessin trace dans ce nouveau clip un rectangle de mêmes dimensions que l'image JPEG. Le nouveau clip masque l'image JPEG en appelant la méthode `MovieClip.setMask()`. Une fois dessiné et configuré, le masque utilise la classe Tween pour animer, c'est-à-dire dévoiler lentement l'image.

3. Enregistrez le document Flash, puis choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

REMARQUE

Pour animer `_alpha` dans l'exemple précédent au lieu de `_yscale`, interpolez directement `target_mc` au lieu du clip de masque.

Pour un exemple de fichier source, drawingapi.fla, qui vous décrit comment utiliser l'API de dessin dans une application Flash, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/DrawingAPI afin d'accéder à cet exemple.

Fonctionnement de la mise à l'échelle et des repères de découpe

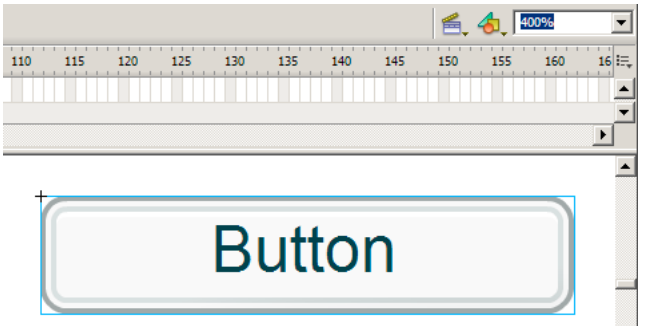
Vous pouvez utiliser la mise à l'échelle à 9 découpes (Echelle-9) pour spécifier l'échelle de style composant des clips. La mise à l'échelle à 9 découpes vous permet de créer des symboles de clip mis à l'échelle de manière appropriée afin d'être utilisés en tant que composants d'interface utilisateur, à la différence du type de redimensionnement généralement appliqué aux graphiques et aux éléments de conception.

Fonctionnement de la mise à l'échelle à 9 découpes

Le plus simple, pour comprendre le fonctionnement de la mise à l'échelle à 9 découpes, est d'observer directement un exemple d'échelle à 9 découpes dans Flash.

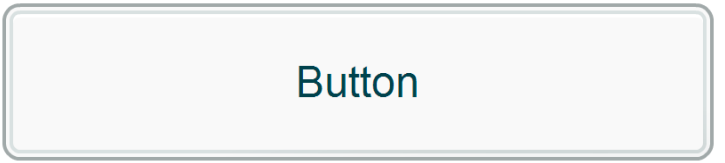
Fonctionnement de la mise à l'échelle 9 dans Flash :

1. Créez un document Flash et enregistrez-le sous le nom de **dynmask.fla**.
2. Faites glisser une copie du composant Button du panneau Composants vers la scène (Fenêtre > Composants).
3. A l'aide de l'outil Zoom, définissez le niveau de zoom de la scène sur 400 %.



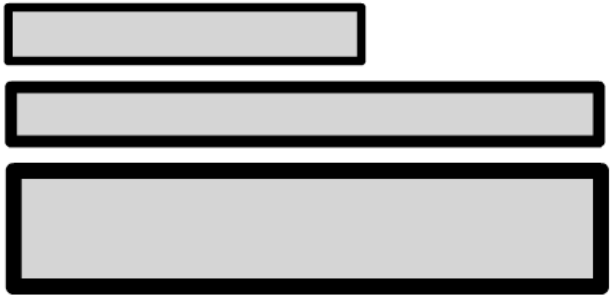
Par défaut, l'occurrence du composant Button présente une largeur de 100 pixels et une hauteur de 22 pixels.

4. A l'aide de l'inspecteur des propriétés, redimensionnez l'occurrence du composant Button à une largeur de 200 pixels et une hauteur de 44 pixels.



Vous constatez que, bien que le composant ait été redimensionné, la bordure et l'étiquette de texte du Bouton ne sont pas déformées. L'étiquette du Bouton est en effet restée au centre et la taille de sa police est inchangée. Quoique les composants de la version2 de l'architecture des composants Adobe n'emploient pas la mise à l'échelle9, les composants manipulent la mise à l'échelle dans la version 2 de l'architecture de composants pour empêcher les contours de changer de taille (comme l'indique la figure suivante).

Imaginez que l'occurrence de bouton est découpée en 9 morceaux distincts, ou en une grille de 3 x 3 morceaux, à l'instar d'un clavier téléphonique ou numérique. Lorsque vous redimensionnez horizontalement l'occurrence de bouton, seuls les trois segments verticaux du centre (les touches 2, 5 et 8 sur un clavier) s'étirent de manière à ce que votre contenu n'apparaisse pas déformé. Lorsque vous redimensionnez verticalement l'occurrence de bouton, seuls les trois segments horizontaux du centre (les touches 4, 5 et 6 sur un clavier) sont redimensionnés. Les quatre coins de la grille ne sont absolument pas mis à l'échelle, ce qui permet au composant de s'agrandir uniformément, sans apparaître étiré (voir les images suivantes).



CONSEIL

Après la transformation avec l'échelle à 9 découpes, des traits sont créés à partir des bords. Il n'y a donc aucune déformation et aucun détail n'est perdu.

Vous pouvez activer des repères de découpe pour la mise à l'échelle 9 dans l'environnement Flash depuis la boîte de dialogue Convertir en symbole ou depuis la boîte de dialogue Propriétés du symbole. La case Activer les repères d'échelle à 9 découpes ne peut être cochée que si vous publiez pour Flash Player 8 et les versions ultérieures et si le comportement est défini sur clip. Les repères d'échelle à 9 découpes n'existent pas dans les versions précédentes de Flash et ne sont pas disponibles si vous créez un bouton ou un symbole graphique. La mise à l'échelle à 9 découpes peut être activée dans ActionScript en définissant la propriété `scale9Grid` sur une occurrence de clip.

Que vous ayez créé vos repères de découpe via l'interface utilisateur ou via ActionScript, vous pouvez tracer la coordonnée *x*, la coordonnée y, la largeur et la hauteur en suivant la propriété `scale9Grid` du clip.

```
trace(my_mc.scale9Grid); // (x=20, y=20, w=120, h=120)
```

Ce fragment de code suit la valeur de l'objet Rectangle utilisée par la propriété `scale9Grid`. Le rectangle a des coordonnées *x* et *y* de 20 pixels, une largeur de 120 pixels et une hauteur de 120 pixels.

Utilisation de la mise à l'échelle à 9 découpes dans ActionScript

Dans l'exemple suivant, vous utilisez les outils de dessin pour tracer un carré de 300 x 300 pixels, redimensionné par la mise à l'échelle 9. Le carré est divisé en neuf petits carrés d'environ 100 x 100 pixels chacun. Lorsque vous redimensionnez le carré, chaque segment qui n'est pas situé dans un coin s'agrandit jusqu'à atteindre la largeur et la hauteur spécifiées.

Utilisation de la mise à l'échelle à 9 découpes avec ActionScript :

1. Créez un document Flash et enregistrez-le sous le nom de **ninescale.fla**.
2. Faites glisser un composant Button dans la bibliothèque du document actuel.
3. Sélectionnez l'outil Rectangle et dessinez sur la scène un carré rouge (300 x 300 pixels) avec un trait noir de 15 pixels.
4. Sélectionnez l'outil Ovale et dessinez sur la scène un cercle violet (50 x 50 pixels) avec un trait noir de 2 pixels.
5. Sélectionnez le cercle violet et faites-le glisser vers le coin supérieur droit du carré rouge précédemment créé.
6. Sélectionnez l'outil Ovale, dessinez un nouveau cercle d'environ 200 x 200 pixels et positionnez-le sur la scène.
7. Sélectionnez le nouveau cercle sur la scène et faites-le glisser de manière à ce que son centre soit situé dans le coin inférieur gauche du carré.
8. Cliquez à côté de l'occurrence de cercle pour désélectionner le cercle.
9. Double-cliquez à nouveau sur le cercle pour le sélectionner et appuyez sur la touche de retour arrière pour supprimer la forme et effacer une partie circulaire du carré.
10. Avec la souris, sélectionnez le carré rouge tout entier et le cercle violet à l'intérieur.
11. Appuyez sur la touche F8 pour convertir la forme en symbole de clip.
12. Attribuez au clip sur la scène le nom d'occurrence **my_mc**.

13. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
import mx.controls.Button;
import flash.geom.Rectangle;

var grid:Rectangle = new Rectangle(100, 100, 100, 100);

var small_button:Button = this.createClassObject(Button, "small_button",
  10, {label:"Small"});
small_button.move(10, 10);
small_button.addEventListener("click", smallHandler);
function smallHandler(eventObj:Object):Void {
    my_mc._width = 100;
    my_mc._height = 100;
}

var large_button:Button = this.createClassObject(Button, "large_button",
  20, {label:"Large"});
large_button.move(120, 10);
large_button.addEventListener("click", largeHandler);
function largeHandler(eventObj:Object):Void {
    my_mc._width = 450;
    my_mc._height = 300;
}

var toggle_button:Button = this.createClassObject(Button,
  "toggle_button", 30, {label:"scale9Grid=OFF", toggle:true,
  selected:false});
toggle_button.move(420, 10);
toggle_button.setSize(120, 22);
toggle_button.addEventListener("click", toggleListener);
function toggleListener(eventObj:Object):Void {
  if (eventObj.target.selected) {
    eventObj.target.label = "scale9Grid=ON";
    my_mc.scale9Grid = grid;
  } else {
    eventObj.target.label = "scale9Grid=OFF";
    my_mc.scale9Grid = undefined;
  }
}
```

Ce code se compose de cinq sections différentes. La première section importe deux classes : mx.controls.Button (la classe du composant Button) et flash.geom.Rectangle. La deuxième section de code crée une occurrence de classe Rectangle et spécifie des coordonnées *x* et *y* de 100 pixels, ainsi qu'une largeur et une hauteur de 100 pixels. Cette occurrence de rectangle est utilisée pour configurer une grille de mise à l'échelle 9 pour la forme de clip créée ultérieurement.

Ensuite, vous créez une occurrence de composant Button et lui attribuez le nom d'occurrence `small_button`. A chaque fois que vous cliquez sur ce bouton, le clip que vous avez précédemment créé se redimensionne à une largeur et une hauteur de 100 pixels. La quatrième section de code crée dynamiquement une nouvelle occurrence de Bouton dénommée `large_button`. A chaque fois que vous cliquez sur celle-ci, le clip est redimensionné à une largeur de 450 pixels et à une hauteur de 300 pixels. La dernière section de code crée une occurrence de bouton, que l'utilisateur peut faire basculer à loisir. Lorsque le bouton est activé, la grille à 9 découpes est appliquée. Lorsque le bouton est désactivé, la grille à 9 découpes est aussi désactivée.

14. Enregistrez le document Flash, puis choisissez Contrôle > Tester l'animation pour tester le fichier SWF.

Cet exemple de code ajoute et positionne trois occurrences du composant Button sur la scène et crée des écouteurs d'événements pour chaque bouton. Si vous cliquez sur le bouton Large tandis que la grille à 9 découpes est désactivée, vous constatez que l'image se déforme et semble s'étirer. Activez la grille à 9 découpes en cliquant sur le bouton bascule et cliquez à nouveau sur le bouton Large. Lorsque la grille à 9 découpes est activée, le cercle dans le coin supérieur gauche n'apparaît plus déformé.

CHAPITRE 13

Création d'interactivité avec ActionScript

13

Dans une animation simple, Flash Player lit les scènes et les images du fichier SWF en ordre séquentiel. Dans un fichier SWF interactif, votre public utilise le clavier et la souris pour atteindre les différentes parties du fichier SWF, pour déplacer des objets, entrer des informations dans des formulaires et effectuer bien d'autres opérations interactives.

Utilisez ActionScript pour créer des scripts qui indiquent à Flash Player l'action à exécuter lorsqu'un événement survient. Certains événements pouvant déclencher un script se produisent lorsque la tête de lecture atteint une image, qu'un clip est chargé ou purgé, ou que l'utilisateur clique sur un bouton ou appuie sur une touche.

Les scripts peuvent prendre la forme d'une commande unique (pour ordonner la fin de la lecture d'un fichier SWF, par exemple) ou d'une série de commandes et d'instructions (pour évaluer une condition avant d'effectuer une action, par exemple). De nombreuses commandes ActionScript sont simples et permettent de créer des contrôles de base pour un fichier SWF. D'autres actions exigent une certaine connaissance des langages de programmation et sont destinées à un développement avancé.

Pour plus d'informations sur la création d'interactivité avec ActionScript, consultez les rubriques suivantes :

Evénements et interaction

Chaque clic de souris ou pression sur une touche génère un événement. Ces types d'événement portent généralement le nom d'*événements utilisateur*, car ils sont générés en réponse à une action de l'utilisateur final. Vous pouvez utiliser ActionScript pour répondre à ces événements ou les *gérer*. Par exemple, lorsqu'un utilisateur clique sur un bouton, il peut être intéressant d'envoyer la tête de lecture vers une autre image du fichier SWF ou de charger une nouvelle page dans le navigateur.

Dans un fichier SWF, les boutons, clips et champs de texte génèrent tous des événements auxquels vous pouvez répondre. ActionScript offre trois manières de gérer les événements : les méthodes des gestionnaires d'événement, les écouteurs d'événements et les gestionnaires `on()` et `onClipEvent()`. Pour plus d'informations sur les événements et leurs gestionnaires, consultez le Chapitre 9, « Gestion d'événements ».

Contrôle de la lecture d'un fichier SWF

Les fonctions ActionScript suivantes vous permettent de contrôler la tête de lecture dans le scénario et de charger une nouvelle page Web dans une fenêtre de navigateur :

- Les fonctions `gotoAndPlay()` et `gotoAndStop()` envoient la tête de lecture vers une image ou une scène. Ce sont des fonctions générales que vous pouvez appeler à partir de tout script. Vous pouvez également utiliser les méthodes `MovieClip.gotoAndPlay()` et `MovieClip.gotoAndStop()` pour naviguer dans le scénario d'un objet clip spécifique. Voir « Déplacement vers une image ou une scène », à la page 559.
- Les actions `play()` et `stop()` entraînent la lecture et l'arrêt des fichiers SWF. Voir « Lecture et arrêt de clips », à la page 559.
- L'action `getURL()` permet d'atteindre une URL différente. Voir « Déplacement vers une URL différente », à la page 560.

Pour plus d'informations, consultez les sections suivantes :

- « Déplacement vers une image ou une scène », à la page 559
- « Lecture et arrêt de clips », à la page 559
- « Déplacement vers une URL différente », à la page 560

Déplacement vers une image ou une scène

Pour atteindre une image ou une scène spécifique du fichier SWF, vous pouvez utiliser les fonctions globales `gotoAndPlay()` ou `gotoAndStop()` ou les méthodes équivalentes `MovieClip.gotoAndPlay()` et `MovieClip.gotoAndStop()` de la classe MovieClip. Chaque fonction ou méthode vous permet de spécifier l'image de la scène en cours à atteindre. Si le document contient plusieurs scènes, vous pouvez en plus spécifier celle que vous souhaitez atteindre.

L'exemple suivant utilise la fonction globale `gotoAndPlay()` figurant dans le gestionnaire d'événements `onRelease` d'un objet bouton pour envoyer à l'image 10 la tête de lecture du scénario contenant le bouton :

```
jump_btn.onRelease = function () {
  gotoAndPlay(10);
};
```

Dans l'exemple suivant, la méthode `MovieClip.gotoAndStop()` envoie le scénario d'une occurrence de clip nommée `categories_mc` à l'image 10 puis s'arrête. Lorsque vous utilisez les méthodes MovieClip `gotoAndPlay()` et `gotoAndStop()`, vous devez indiquer l'occurrence à laquelle elles s'appliquent.

```
jump_btn.onPress = function () {
  categories_mc.gotoAndStop(10);
};
```

Dans le dernier exemple, la fonction `gotoAndStop()` est utilisée pour déplacer la tête de lecture de l'image 1 de la séquence 2. Si aucune séquence n'est spécifiée, la tête de lecture passe à l'image spécifiée dans la séquence en cours. Le paramètre séquence est réservé au scénario racine. Vous ne pouvez pas l'utiliser dans les scénarios des clips ou autres objets du document.

```
nextScene_mc.onRelease = function() {
  gotoAndStop("Scene 2", 1);
}
```

Lecture et arrêt de clips

Sauf indication contraire, une fois démarré, un fichier SWF exécute toutes les images du scénario. Vous pouvez démarrer ou arrêter un fichier SWF à l'aide des fonctions globales `play()` et `stop()` ou des méthodes MovieClip équivalentes. Ainsi, la fonction ou la méthode `stop()` vous permet d'arrêter un fichier SWF à la fin d'une scène avant de passer à la scène suivante. Une fois un fichier SWF arrêté, il doit être redémarré au moyen de `play()` ou `gotoAndPlay()`.

Vous pouvez utiliser les fonctions `play()` et `stop()` ou les méthodes MovieClip pour contrôler le scénario principal ou le scénario de tout clip ou de tout fichier SWF chargé. Le clip que vous souhaitez contrôler doit posséder un nom d'occurrence et être présent dans le scénario.

Le gestionnaire `on(press)` suivant, lorsqu'il est associé à un bouton, lance la tête de lecture dans le fichier SWF ou le clip contenant l'objet bouton :

```
// Associé à une occurrence de bouton
on (press) {
  // Lit le scénario contenant le bouton
  play();
}
```

Ce même gestionnaire d'événements `on()` donnerait un résultat différent s'il était associé à un clip plutôt qu'à un bouton. Lorsqu'elles sont associées à un objet bouton, les instructions faites dans un gestionnaire `on()` sont appliquées par défaut au scénario contenant le bouton. En revanche, lorsqu'elles sont associées à un clip, les instructions définies dans un gestionnaire `on()` sont appliquées au clip contenant le gestionnaire.

Le gestionnaire `onPress()` suivant, par exemple, arrête le scénario du clip auquel il est associé, et non le scénario contenant le clip :

```
// Associé à l'occurrence de clip myMovie_mc
myMovie_mc.onPress() {
  stop();
};
```

Il en va de même pour les gestionnaires `onClipEvent()` associés à des clips. Le code suivant, par exemple, arrête le scénario du clip contenant le gestionnaire `onClipEvent()` lors du premier chargement du clip ou de sa première apparition sur la scène :

```
onClipEvent(load) {
  stop();
}
```

Déplacement vers une URL différente

Pour ouvrir une page Web dans une fenêtre de navigateur ou pour transmettre des données à une autre application sur une URL définie, utilisez la fonction globale `getURL()` ou la méthode `MovieClip.getURL()`. Vous pouvez, par exemple, lier un bouton à un nouveau site Web ou envoyer des variables de scénario à un script CGI pour traitement comme s'il s'agissait d'un formulaire HTML. Vous pouvez également spécifier une fenêtre cible, exactement comme pour le ciblage d'une fenêtre à l'aide d'une balise d'ancrage HTML (`<a></a>`).

Par exemple, le code suivant ouvre la page d'accueil adobe.com dans une fenêtre de navigateur vide lorsque l'utilisateur clique sur l'occurrence de bouton `homepage_btn` :

```
// Associer à une image
homepage_btn.onRelease = function () {
  getURL("http://www.adobe.com", "_blank");
};
```

Vous pouvez également envoyer des variables avec l'URL, en utilisant les méthodes `GET` ou `POST`. Cette opération est utile si la page que vous chargez provient d'un serveur d'applications, telle une page ColdFusion Server (CFM), et attend la réception de variables. Supposons par exemple que vous souhaitiez charger une page CFM nommée addUser.cfm qui attend deux variables de formulaire, `firstName` et `age`. Pour ce faire, vous pouvez créer un clip nommé `variables_mc` qui définit ces deux variables, comme indiqué dans l'exemple suivant :

```
variables_mc.firstName = "Francois";
variables_mc.age = 32;
```

Le code suivant charge ensuite addUser.cfm dans une fenêtre de navigateur vide, puis transmet `variables_mc.name` et `variables_mc.age` à la page CFM dans l'en-tête `POST` :

```
variables_mc.getURL("addUser.cfm", "_blank", "POST");
```

Les fonctionnalités de `getURL()` sont fonction du navigateur que vous utilisez. Le procédé le plus fiable pour harmoniser le fonctionnement de tous les navigateurs est d'appeler une fonction JavaScript dans le code HTML utilisant la méthode JavaScript `window.open()` pour ouvrir une fenêtre. Ajoutez à votre modèle HTML les HTML et JavaScript suivants :

```
<script language="JavaScript">
<--
  function openNewWindow(myURL) {
    window.open(myURL, "targetWindow");
  }
// -->
</script>
```

Vous pouvez utiliser l'ActionScript suivant pour appeler`openNewWindow` à partir de votre fichier SWF :

```
var myURL:String = "http://foo.com";
getURL("javascript:openNewWindow('" + String(myURL) + "');");
```

Pour plus d'informations, consultez la section relative à la fonction getURL du *Guide de référence du langage ActionScript 2.0*.

Création de l'interactivité et des effets visuels

Pour ajouter de l'interactivité et d'autres effets visuels, vous devez connaître les techniques suivantes :

Création d'un pointeur de souris personnalisé

Un pointeur de souris standard est la représentation visuelle par le système d'exploitation de la position de la souris de l'utilisateur. Si vous remplacez le pointeur standard par un pointeur conçu dans Flash, vous pouvez intégrer plus étroitement les mouvements de la souris de l'utilisateur dans le fichier SWF. L'exemple de cette section utilise un pointeur personnalisé qui ressemble à une grande flèche. Cependant, la puissance de cette fonctionnalité réside dans la possibilité de donner au pointeur n'importe quelle apparence (par exemple, un ballon de football qui doit franchir la ligne de but ou un rouleau de tissu que l'utilisateur tire au-dessus d'un fauteuil afin de modifier sa couleur).

Pour créer un pointeur personnalisé, vous devez créer son clip sur la scène. Ensuite, dans ActionScript, masquez le pointeur standard et associez le clip aux mouvements du curseur personnalisé. Pour masquer le pointeur standard, utilisez la méthode `hide()` de la classe Mouse intégrée (consultez la section hide (méthode Mouse.hide) dans le *Guide de référence du langage ActionScript 2.0*).

Pour créer un pointeur personnalisé :

1. Créez un clip à utiliser comme pointeur personnalisé et placez une occurrence du clip sur la scène.
2. Sélectionnez l'occurrence de clip sur la scène.
3. Dans l'inspecteur Propriétés, tapez **cursor_mc** dans la zone de texte Nom de l'occurrence.

4. Sélectionnez l'image 1 du scénario, puis tapez le code suivant dans le panneau Actions :

```
Mouse.hide();
cursor_mc.onMouseMove = function() {
  this._x = _xmouse;
  this._y = _ymouse;
  updateAfterEvent();
};
```

   La méthode `Mouse.hide()` masque le pointeur lorsque le clip apparaît pour la première fois sur la scène. La fonction `onMouseMove` place le pointeur personnalisé au même endroit que le pointeur standard et appelle `updateAfterEvent()` chaque fois que l'utilisateur déplace la souris.

   La fonction `updateAfterEvent()` actualise l'écran dès que l'événement spécifié s'est produit, sans attendre le dessin de l'image suivante, ce qui correspond au comportement par défaut. (Consultez la section relative à la fonction updateAfterEvent dans le *Guide de référence du langage ActionScript 2.0*.)

5. Sélectionnez Contrôle > Tester l'animation pour tester votre pointeur personnalisé.

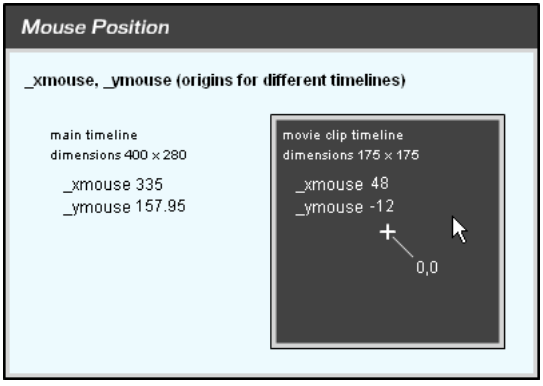
Les boutons fonctionnent toujours avec un pointeur de souris personnalisé. Il est conseillé de placer le pointeur personnalisé sur le calque supérieur du scénario, de sorte qu'il s'affiche au premier plan, par-dessus les boutons et autres objets dans les autres calques, lorsque vous déplacez la souris dans le fichier SWF. De plus, l'extrémité d'un pointeur personnalisé est le point d'alignement du clip que vous utilisez comme pointeur. Pour qu'elle corresponde à une partie spécifique du clip, définissez les coordonnées du point d'alignement de façon à ce qu'ils coïncident avec ce point.

Pour plus d'informations sur les méthodes de la classe Mouse, consultez la section Mouse dans le *Guide de référence du langage ActionScript 2.0*.

Obtention de la position de la souris

Vous pouvez utiliser les propriétés `_xmouse` et `_ymouse` pour déterminer la position du pointeur de la souris dans un fichier SWF. Ces propriétés peuvent être utilisées, par exemple, dans une application cartographique qui reprend les valeurs des propriétés `_xmouse` et `_ymouse`, et les utilise pour calculer la longitude et la latitude d'un emplacement spécifique.

Chaque scénario possède une propriété _xmouse et _ymouse qui renvoie l'emplacement du pointeur dans son système de coordonnées. La position est toujours donnée par rapport au point d'alignement. Dans le scénario principal (_level0), le point d'alignement se trouve dans le coin supérieur gauche. Pour un clip, le point d'alignement dépend du point d'alignement défini lorsque le clip a été créé ou lors de son placement sur la scène.



Les propriétés _xmouse et _ymouse dans le scénario principal et dans le scénario d'un clip

La procédure suivante montre plusieurs façons d'obtenir la position du pointeur dans le scénario principal ou dans un clip.

Pour obtenir la position actuelle du pointeur :

1. Créez deux champs texte dynamiques et nommez-les **box1_txt** et **box2_txt**.
2. Ajoutez des étiquettes pour les zones de texte : coordonnées *x* et *y*, respectivement.
3. Sélectionnez Fenêtre > Actions pour ouvrir le panneau Actions, si ce dernier n'est pas déjà ouvert.
4. Ajoutez le code suivant au fichier :

```
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  // renvoie les coordonnées X et Y de la souris
  box1_txt.text = _xmouse;
  box2_txt.text = _ymouse;
};
Mouse.addListener(mouseListener);
```

5. Sélectionnez Contrôle > Tester l'animation pour tester le clip Flash. Les champs box1_txt et box2_txt affichent la position du pointeur lorsque vous le déplacez au-dessus de la scène.

Pour plus d'informations sur les propriétés _xmouse et _ymouse, consultez les sections _xmouse (propriété MovieClip._xmouse) et _ymouse (propriété MovieClip._ymouse) dans le *Guide de référence du langage ActionScript 2.0*.

Capture des pressions sur les touches

Vous pouvez utiliser le gestionnaire global `on()` pour intercepter le comportement intégré des pressions sur les touches dans Flash Player, comme il est indiqué dans l'exemple suivant :

```
/* Lorsque vous appuyez sur la touche Flèche vers la gauche ou la droite, la
  transparence du clip associé au gestionnaire change. */
on (keyPress "<Left>") {
  this._alpha -= 10;
}
on (keyPress "<Right>") {
  this._alpha += 10;
}
```

Assurez-vous que Contrôle > Désactiver les raccourcis clavier est sélectionnée ou que les touches associées à un comportement intégré ne seront pas ignorées lorsque vous testez l'application avec Contrôle > Tester l'animation. Consultez le paramètre `keyPress` du gestionnaire on dans le *Guide de référence du langage ActionScript 2.0*.

Vous pouvez utiliser les méthodes de la classe intégrée Key pour détecter la dernière touche sur laquelle l'utilisateur a appuyé. La classe Key ne requiert aucune fonction constructeur ; appelez ses méthodes sur la classe même, comme illustré dans l'exemple suivant :

```
Key.getCode();
```

Vous pouvez obtenir les codes virtuels ou les valeurs ASCII (American Standard Code for Information Interchange) des touches :

- Pour obtenir le code réel de la dernière touche enfoncée par l'utilisateur, utilisez la méthode `getCode()`.
- Pour obtenir la valeur ASCII de la dernière touche enfoncée par l'utilisateur, utilisez la méthode `getAscii()`.

Un code est affecté à chaque touche physique du clavier. Ainsi, la touche Flèche vers la gauche est associée au code virtuel 37. L'utilisation de ces codes vous permet de vous assurer que les commandes de votre fichier SWF sont les mêmes sur tous les claviers, quelle que soit la langue ou la plate-forme utilisée.

Des valeurs ASCII sont affectées aux 127 premiers caractères de chaque jeu de caractères. Les valeurs ASCII fournissent des informations sur un caractère affiché à l'écran. Par exemple, les lettres « A » et « a » ont des valeurs ASCII différentes.

Pour décider des touches que vous allez utiliser et déterminer leurs codes, utilisez l'une des approches suivantes :

- La liste des codes de touches est publiée dans l'Annexe C, « Touches du clavier et valeurs de code correspondantes »
- Utilisez une constante de classe Key. (Dans la boîte à outils Actions, cliquez sur Classes ActionScript 2.0 > Animation > Animation > Constantes.)

- Affectez le gestionnaire `onClipEvent()` suivant à un clip, puis choisissez Contrôle > Tester l'animation et appuyez sur la touche souhaitée :

```
onClipEvent(keyDown) {
  trace(Key.getCode());
}
```

Le code de la touche désirée s'affiche dans le panneau de sortie.

Les méthodes de classe Key sont fréquemment placées dans un gestionnaire d'événement. Dans l'exemple suivant, l'utilisateur déplace la voiture à l'aide des touches de direction. La méthode `Key.isDown()` indique si la touche enfoncée est la flèche orientée vers la droite, la gauche, le haut ou le bas. L'écouteur d'événement, `Key.onKeyDown`, détermine la valeur de `Key.isDown(keyCode)` à partir des instructions `if`. En fonction de la valeur, il demande à Flash Player de mettre à jour la position de la voiture et d'afficher la direction.

L'exemple suivant indique comment identifier les touches utilisées pour déplacer un clip vers le haut, le bas, la gauche ou la droite de la scène en fonction de la touche de direction utilisée (haut, bas, gauche ou droite). Par ailleurs, un champ de texte indique le nom de la touche enfoncée.

Pour créer un clip activé via le clavier :

1. Sur la scène, créez un clip qui se déplacera en réponse aux touches de direction du clavier.

   Dans cet exemple, le nom de l'occurrence de clip est `car_mc`.
2. Sélectionnez l'image 1 dans le scénario et, le cas échéant, sélectionnez Fenêtre > Actions pour ouvrir le panneau Actions.
3. Pour déterminer la distance parcourue par la voiture à l'écran après chaque pression sur une touche, définissez une variable `distance` et fixez sa valeur à 10 :

   ```
   var distance:Number = 10;
   ```
4. Ajoutez le code ActionScript suivant au panneau Actions, après le code existant :

   ```
   this.createTextField("display_txt", 999, 0, 0, 100, 20);
   ```
5. Pour créer le gestionnaire d'événements du clip voiture qui vérifie quelle touche de direction (gauche, droite, bas ou haut) est enfoncée, ajoutez le code suivant au panneau Actions :

   ```
   var keyListener:Object = new Object();
   keyListener.onKeyDown = function() {
   };
   Key.addListener(keyListener);
   ```

6. Pour vérifier si la touche de direction vers la droite est enfoncée et pour déplacer le clip voiture en conséquence, ajoutez un code au corps du gestionnaire d'événements `onEnterFrame`.

   Votre code doit alors ressembler à l'exemple suivant (le nouveau code apparaît en gras) :

```
var distance:Number = 10;
this.createTextField("display_txt", 999, 0, 0, 100, 20);
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.LEFT)) {
    car_mc._x = Math.max(car_mc._x - distance, 0);
    display_txt.text = "Left";
  }
};
Key.addListener(keyListener);
```

   Si vous appuyez sur la touche de direction vers la gauche, la propriété _x de la voiture est définie sur la valeur actuelle de _x moins la distance ou la valeur 0, en retenant la valeur la plus élevée. Par conséquent, la valeur de la propriété _x n'est jamais négative. En outre, le mot *Left* doit s'afficher dans le fichier SWF.

7. Adaptez ce code pour vérifier si les autres touches de direction (droite, haut ou bas) ont été enfoncées.

   Votre code doit alors ressembler à l'exemple suivant (le nouveau code apparaît en gras) :

```
var distance:Number = 10;
this.createTextField("display_txt", 999, 0, 0, 100, 20);
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.LEFT)) {
    car_mc._x = Math.max(car_mc._x - distance, 0);
    display_txt.text = "Left";
  } else if (Key.isDown(Key.RIGHT)) {
    car_mc._x = Math.min(car_mc._x + distance, Stage.width -
  car_mc._width);
    display_txt.text = "Right";
  } else if (Key.isDown(Key.UP)) {
    car_mc._y = Math.max(car_mc._y - distance, 0);
    display_txt.text = "Up";
  } else if (Key.isDown(Key.DOWN)) {
    car_mc._y = Math.min(car_mc._y + distance, Stage.height -
  car_mc._height);
    display_txt.text = "Down";
  }
};
Key.addListener(keyListener);
```

8. Sélectionnez Contrôle > Tester l'animation pour tester le fichier.

Pour plus d'informations sur les méthodes de la classe Key, consultez la section Key dans le *Guide de référence du langage ActionScript 2.0*.

Définition des valeurs de couleurs

Vous pouvez utiliser les méthodes de la classe ColorTransform intégrée (flash.geom.ColorTransform) pour régler la couleur d'un clip. La méthode `rgb` de la classe ColorTransform affecte des valeurs rouge, vert, bleu (RVB) hexadécimales au clip. L'exemple suivant utilise `rgb` pour changer la couleur d'un objet en réponse aux actions de l'utilisateur.

Pour définir la valeur de couleur d'un clip :

1. Créez un nouveau document Flash, puis enregistrez-le sous le nom **setrgb.fla**.
2. A l'aide de l'outil Rectangle, dessinez un grand carré sur la scène.
3. Convertissez la forme du symbole d'un clip et donnez au symbole le nom d'occurrence de **car_mc** dans l'inspecteur Propriétés.
4. Créez un symbole de bouton appelé colorChip, placez quatre occurrences du bouton sur la scène et nommez-les `red_btn`, `green_btn`, `blue_btn` et `black_btn`.
5. Sélectionnez l'image 1 dans le scénario principal, puis sélectionnez la Fenêtre > Actions.
6. Ajoutez le code suivant à l'image 1 du scénario principal :

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
var trans:Transform = new Transform(car_mc);
trans.colorTransform = colorTrans;
```

7. Pour que le bouton bleu colore le clip `car_mc` en bleu, ajoutez le code suivant au panneau Actions :

```
blue_btn.onRelease = function() {
  colorTrans.rgb = 0x333399; // blue
  trans.colorTransform = colorTrans;
};
```

   Le fragment de code précédent change la propriété `rgb` de l'objet de transformation de couleurs et applique nouveau l'effet de transformation de couleurs au clip `car_mc` à chaque pression du bouton.

8. Répétez l'étape 7 pour les autres boutons (`red_btn`, `green_btn`, et `black_btn`) pour changer la couleur du clip à la couleur correspondante.

   Votre code doit alors ressembler à l'exemple suivant (le nouveau code apparaît en gras) :

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
var trans:Transform = new Transform(car_mc);
trans.colorTransform = colorTrans;

blue_btn.onRelease = function() {
  colorTrans.rgb = 0x333399; // blue
  trans.colorTransform = colorTrans;
};
red_btn.onRelease = function() {
  colorTrans.rgb = 0xFF0000; // red (rouge)
  trans.colorTransform = colorTrans;
};
green_btn.onRelease = function() {
  colorTrans.rgb = 0x006600; // green (vert)
  trans.colorTransform = colorTrans;
};
black_btn.onRelease = function() {
  colorTrans.rgb = 0x000000; // black (noir)
  trans.colorTransform = colorTrans;
};
```

9. Sélectionnez Contrôle > Tester l'animation pour modifier la couleur du clip.

Pour plus d'informations sur les méthodes de la classe ColorTransform, consultez la section ColorTransform (flash.geom.ColorTransform) dans le *Guide de référence du langage ActionScript 2.0*.

Création de commandes audio

La classe Sound intégrée permet de contrôler les sons d'un fichier SWF. Pour utiliser les méthodes de la classe Sound, vous devez d'abord créer un objet Sound. Faites ensuite appel à la méthode `attachSound()` pour insérer un son de la bibliothèque dans un fichier SWF pendant sa lecture.

La méthode `setVolume()` de la classe Sound contrôle le volume et la méthode `setPan()` règle la balance gauche et droite d'un son.

Les procédures suivantes expliquent comment créer des contrôles audio.

Pour associer un son à un scénario :

1. Sélectionnez Fichier > Importer > Importer dans la bibliothèque pour importer un son.
2. Sélectionnez le son dans la bibliothèque, puis cliquez dessus avec le bouton droit de la souris (Windows) ou en appuyant sur la touche Contrôle (Macintosh), et choisissez Liaison.
3. Activez les options Exporter pour ActionScript et Exporter dans la première image, puis affectez au son l'identifiant `a_thousand_ways`.
4. Ajoutez un bouton sur la scène et appelez-le `play_btn`.
5. Ajoutez un autre bouton sur la scène et appelez-le `stop_btn`.
6. Sélectionnez l'image 1 dans le scénario principal, puis sélectionnez la Fenêtre > Actions.

   Ajoutez le code suivant au panneau Actions :

```
var song_sound:Sound = new Sound();
song_sound.attachSound("a_thousand_ways");
play_btn.onRelease = function() {
  song_sound.start();
};
stop_btn.onRelease = function() {
  song_sound.stop();
};
```

   Ce code commence par arrêter le clip du haut-parleur. Il crée ensuite un objet Sound (`song_sound`) et lui associe le son dont l'identifiant de liaison correspond à `a_thousand_ways`. Enfin, les gestionnaires d'événement `onRelease` associés aux objets `play_btn` et `stop_btn` démarrent et arrêtent le son à l'aide des méthodes `Sound.start()` et `Sound.stop()`. Ils permettent également de lire et d'arrêter le son joint.
7. Sélectionnez Contrôle > Tester l'animation pour entendre le son.

Pour créer une commande de volume réglable :

1. A l'aide de l'outil Rectangle, dessinez sur la scène un petit rectangle, d'environ 100pixels de large et 20pixels de haut.
2. Sélectionnez l'outil Sélection et double-cliquez sur la forme de la scène.
3. Appuyez sur F8 pour ouvrir la boîte de dialogue Convertir en symbole.
4. Sélectionnez le type de bouton, tapez un nom de symbole pour **volume** et cliquez sur OK.
5. Une fois le symbole de bouton sélectionné sur la scène, tapez le nom d'occurrence de **handle_btn** dans l'inspecteur Propriétés.
6. Sélectionnez le bouton et sélectionnez Modification > Convertir en symbole.

   N'oubliez pas de sélectionner le comportement du clip. Cela crée un clip avec le bouton sur l'image 1.

7. Sélectionnez le clip et entrez **volume_mc** en tant que nom d'occurrence dans l'inspecteur Propriétés.
8. Sélectionnez l'image 1 du scénario principal, puis sélectionnez la Fenêtre > Actions.
9. Entrez le code suivant dans le panneau Actions :

```
this.createTextField("volume_txt", 10, 30, 30, 200, 20);
volume_mc.top = volume_mc._y;
volume_mc.bottom = volume_mc._y;
volume_mc.left = volume_mc._x;
volume_mc.right = volume_mc._x + 100;
volume_mc._x += 100;

volume_mc.handle_btn.onPress = function() {
  startDrag(this._parent, false, this._parent.left, this._parent.top,
  this._parent.right, this._parent.bottom);
};
volume_mc.handle_btn.onRelease = function() {
  stopDrag();
  var level:Number = Math.ceil(this._parent._x - this._parent.left);
  this._parent._parent.song_sound.setVolume(level);
  this._parent._parent.volume_txt.text = level;
};
volume_mc.handle_btn.onReleaseOutside = slider_mc.handle_btn.onRelease;
```

   Les paramètres de `startDrag()` `gauche`, `haut`, `droite` et `bas` sont des variables définies dans une action de clip.
10. Sélectionnez Contrôle > Tester l'animation pour utiliser la commande de volume.

Pour créer une commande de balance réglable :

1. A l'aide de l'outil Rectangle, dessinez sur la scène un petit rectangle, d'environ 100pixels de large et 20pixels de haut.
2. Sélectionnez l'outil Sélection et double-cliquez sur la forme de la scène.
3. Appuyez sur F8 pour lancer la boîte de dialogue Convertir en symbole.
4. Sélectionnez le type de bouton, tapez un nom de symbole pour **balance** et cliquez sur OK.
5. Une fois le symbole de bouton sélectionné sur la scène, tapez un nom d'occurrence pour **handle_btn** dans l'inspecteur Propriétés.
6. Sélectionnez le bouton et sélectionnez Modification > Convertir en symbole.

   N'oubliez pas de sélectionner le comportement du clip. Cela crée un clip avec le bouton sur l'image 1.
7. Sélectionnez le clip et entrez **balance_mc** en tant que nom d'occurrence dans l'inspecteur Propriétés.

8. Entrez le code suivant dans le panneau Actions :

```
balance_mc.top = balance_mc._y;
balance_mc.bottom = balance_mc._y;
balance_mc.left = balance_mc._x;
balance_mc.right = balance_mc._x + 100;
balance_mc._x += 50;
balance_mc.handle_btn.onPress = function() {
  startDrag(this._parent, false, this._parent.left, this._parent.top,
  this._parent.right, this._parent.bottom);
};
balance_mc.handle_btn.onRelease = function() {
  stopDrag();
  var level:Number = Math.ceil((this._parent._x - this._parent.left -
  50) * 2);
  this._parent._parent.song_sound.setPan(level);
};
balance_mc.handle_btn.onReleaseOutside =
  balance_mc.handle_btn.onRelease;
```

Les paramètres de `startDrag()` `gauche`, `haut`, `droite` et `bas` sont des variables définies dans une action de clip.

9. Sélectionnez Contrôle > Tester l'animation pour utiliser le curseur de balance.

Pour plus d'informations sur les méthodes de la classe Sound, consultez la section Sound dans le *Guide de référence du langage ActionScript 2.0*.

Détection des collisions

La méthode `hitTest()` de la classe MovieClip détecte les collisions dans un fichier SWF. Elle vérifie si un objet est entré en collision avec un clip et renvoie une valeur booléenne (`true` ou `false`).

Il peut être utile de savoir si une collision s'est produite, soit pour tester si l'utilisateur a atteint une zone statique précise de la scène, soit pour déterminer si un clip en a atteint un autre. La méthode `hitTest()` permet de déterminer ces résultats.

Vous pouvez utiliser les paramètres de `hitTest()` pour définir les coordonnées *x* et *y* d'une zone réactive sur la scène ou utiliser le chemin cible d'un autre clip comme zone réactive. Lorsque vous définissez *x* et *y*, `hitTest()` renvoie la valeur `true` si le point identifié par (*x*,*y*) n'est pas transparent. Lorsqu'une cible est transmise à `hitTest()`, les cadres de délimitation des deux clips sont comparés. S'ils se recoupent, `hitTest()` renvoie la valeur `true`. S'ils ne se croisent pas, `hitTest()` renvoie la valeur `false`.

Vous pouvez également utiliser `hitTest()` pour tester une collision entre deux clips.

L'exemple suivant illustre la détection d'une collision entre la souris et les clips de la scène.

Pour détecter une collision entre un clip et le curseur de la souris :

1. Dans le scénario, sélectionnez la première image du calque 1.
2. Sélectionnez Fenêtre > Actions pour ouvrir le panneau Actions, si ce dernier n'est pas déjà ouvert.
3. Entrez le code suivant dans le panneau Actions :

```
this.createEmptyMovieClip("box_mc", 10);
with (box_mc) {
  beginFill(0xFF0000, 100);
  moveTo(100, 100);
  lineTo(200, 100);
  lineTo(200, 200);
  lineTo(100, 200);
  lineTo(100, 100);
  endFill();
}

this.createTextField("status_txt", 999, 0, 0, 100, 22);

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  status_txt.text = _level0.hitTest(_xmouse, _ymouse, true);
}
Mouse.addListener(mouseListener);
```

4. Sélectionnez Contrôle > Tester l'animation et passez la souris sur le clip pour tester la collision.

   La valeur `true` lorsque le pointeur est placé sur un pixel non transparent.

Pour détecter la collision entre deux clips :

1. Faites glisser deux clips jusqu'à la scène et affectez-leur les noms d'occurrence `car_mc` et `area_mc`.
2. Sélectionnez l'image 1 du scénario.
3. Sélectionnez Fenêtre > Actions pour ouvrir le panneau Actions, si ce dernier n'est pas déjà visible.

4. Entrez le code suivant dans le panneau Actions :

```
this.createTextField("status_txt", 999, 10, 10, 100, 22);
area_mc.onEnterFrame = function() {
  status_txt.text = this.hitTest(car_mc);
};

car_mc.onPress = function() {
  this.startDrag(false);
  updateAfterEvent();
};
car_mc.onRelease = function() {
  this.stopDrag();
};
```

5. Sélectionnez Contrôle > Tester l'animation et faites glisser le clip pour tester la collision.

   Lorsque le cadre de délimitation de la voiture touche celui de la zone, l'état devient `true`.

Pour plus d'informations, consultez la section hitTest (méthode MovieClip.hitTest) dans le *Guide de référence du langage ActionScript 2.0*.

Création d'un outil de dessin de ligne simple

Vous pouvez utiliser les méthodes de la classe MovieClip pour dessiner des formes et des remplissages sur la scène en cours de lecture. Vous pouvez ainsi créer des outils de dessin pour les utilisateurs et tracer des formes dans le fichier SWF en réponse à des événements.
Les méthodes de dessin sont `beginFill()`, `beginGradientFill()`, `clear()`, `curveTo()`, `endFill()`, `lineTo()`, `lineStyle()` et `moveTo()`.

Vous pouvez appliquer ces méthodes à toute occurrence de clip (par exemple, `myClip.lineTo()`) ou à un niveau (`_level0.curveTo()`).

Les méthodes `lineTo()` et `curveTo()` vous permettent respectivement de dessiner des lignes et des courbes. La méthode `lineStyle()` permet de spécifier une couleur et une épaisseur de ligne et un paramètre alpha pour une ligne ou une courbe. La méthode de dessin `moveTo()` place la position de dessin actuelle sur les coordonnées de scène, *x* et *y*, spécifiées.

Les méthodes `beginFill()` et `beginGradientFill()` remplissent respectivement un chemin fermé avec une couleur de remplissage unie ou dégradée, tandis que `endFill()` applique le remplissage spécifié dans le dernier appel à `beginFill()` ou `beginGradientFill()`.
La méthode `clear()` efface ce qui a été dessiné dans l'objet clip spécifié.

Pour créer un outil de dessin de ligne simple :

1. Dans un nouveau document, créez un bouton sur la scène et entrez `clear_btn` comme nom d'occurrence dans l'inspecteur Propriétés.
2. Sélectionnez l'image 1 dans le scénario.
3. Sélectionnez Fenêtre > Actions pour ouvrir le panneau Actions, si ce dernier n'est pas déjà visible.
4. Dans le panneau Actions, entrez le code suivant :

```
this.createEmptyMovieClip("canvas_mc", 999);
var isDrawing:Boolean = false;
//
clear_btn.onRelease = function() {
  canvas_mc.clear();
};
//
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  canvas_mc.lineStyle(5, 0xFF0000, 100);
  canvas_mc.moveTo(_xmouse, _ymouse);
  isDrawing = true;
};
mouseListener.onMouseMove = function() {
  if (isDrawing) {
    canvas_mc.lineTo(_xmouse, _ymouse);
    updateAfterEvent();
  }
};
mouseListener.onMouseUp = function() {
  isDrawing = false;
};
Mouse.addListener(mouseListener);
```

5. Sélectionnez Contrôle > Tester l'animation pour tester le document.
6. Faites glisser la souris pour dessiner une ligne sur la scène.
7. Cliquez sur le bouton pour effacer ce que vous avez dessiné.

Création de liaisons de données à l'exécution à l'aide d'ActionScript

Si vous utilisez des composants pour créer des applications, il est souvent nécessaire d'ajouter des liaisons entre ces composants de manière à pouvoir interagir avec les données ou faire en sorte que les composants interagissent entre eux. L'interaction entre les composants est nécessaire pour créer des formulaires ou des interfaces sur lesquels vos utilisateurs peuvent agir. Pour ajouter des liaisons entre des composants sur la scène, utilisez l'onglet Liaisons de l'Inspecteur de composants.

Pour plus d'informations sur l'utilisation de l'onglet Liaisons, consultez le guide *Utilisation de Flash*. Des informations supplémentaires sont également disponibles dans les articles en ligne suivants : Building a Tip of the day Application (Part 2), Data Binding in Macromedia Flash MX Professional 2004 et Building a Google Search Application with Macromedia Flash MX Professional.

Vous pouvez utiliser ActionScript au lieu de passer par l'onglet Liaisons pour créer des liaisons entre des composants. L'ajout de code se révèle souvent plus rapide et plus efficace que l'environnement de programmation. La création de liaisons par du code ActionScript est inévitable lorsque vous utilisez du code pour ajouter des composants à une application. Vous pouvez choisir d'ajouter des composants sur la scène dynamiquement, via la méthode `createClassObject()`. Cependant, il est préférable de ne pas passer par l'onglet Liaisons pour créer une liaison car les composants n'existent pas avant l'exécution. L'ajout d'une liaison de données à l'aide de code ActionScript est souvent appelé *liaison de données à l'exécution*.

Pour plus d'informations, consultez les sections suivantes :

- Création de liaisons entre des composants d'interface à l'aide d'ActionScript
- « Utilisation de composants, de liaisons et de mises en forme personnalisées », à la page 581
- « Ajout et liaison de composants sur la scène », à la page 584

Création de liaisons entre des composants d'interface à l'aide d'ActionScript

Relier les données de deux composants au moment de l'exécution n'est pas une opération complexe. Vous ne devez pas oublier d'inclure le composant DataBindingClasses dans votre document car ce composant contient les classes avec lesquelles vous devez travailler.

Pour créer une liaison entre deux composants TextInput à l'aide d'ActionScript :

1. Créez un nouveau document Flash appelé **panel_as.fla**.
2. Faites glisser deux copies du composant TextInput sur la scène.
3. Donnez les noms d'occurrences suivants aux composants : **in_ti** et **out_ti**.
4. Sélectionnez la Fenêtre > Bibliothèques communes > Classes puis ouvrez la nouvelle bibliothèque commune appelée Classes.fla.
5. Faites glisser une copie du composant DataBindingClasses sur le panneau Bibliothèque ou faites glisser le composant sur la scène, puis supprimez-le.

   Lorsque vous avez terminé, vous pouvez fermer la bibliothèque commune. Lorsque le composant DataBindingClasses est supprimé de la scène, Flash en conserve une copie dans la bibliothèque.

CONSEIL	Si vous oubliez de supprimer le composant DataBindingClasses sur la scène, son icône demeure visible au moment de l'exécution.

REMARQUE	Lorsque vous créez une liaison à l'aide de l'inspecteur de composants, comme dans l'exemple précédent, Flash ajoute automatiquement le composant DataBindingClasses au fichier FLA. Par contre, si vous créez des liaisons de données avec ActionScript, vous devez vous charger de copier cette classe vous-même dans votre bibliothèque, comme l'illustre l'étape suivante.

6. Insérez un nouveau calque et nommez-le **actions**.
7. Ajoutez le code ActionScript suivant dans l'image 1 du calque actions :

```
var src:mx.data.binding.EndPoint = new mx.data.binding.EndPoint();
src.component = in_ti;
src.property = "text";
src.event = "focusOut";
var dest:mx.data.binding.EndPoint = new mx.data.binding.EndPoint();
dest.component = out_ti;
dest.property = "text";
new mx.data.binding.Binding(src, dest);
```

Si vous préférez la version abrégée, vous pouvez importer les classes de liaison et utiliser le code suivant à la place :

```
import mx.data.binding.*;
var src:EndPoint = new EndPoint();
src.component = in_ti;
src.property = "text";
src.event = "focusOut";
var dest:EndPoint = new EndPoint();
dest.component = out_ti;
dest.property = "text";
new Binding(src, dest);
```

Ce code ActionScript crée deux points de terminaison de liaison des données, un pour chaque composant relié. Le premier point de terminaison créé définit le composant auquel il est relié (`in_ti`), la propriété à rechercher (`text`), et l'événement qui doit déclencher la liaison (`focusOut`). Le second point de terminaison ne contient que le composant et la propriété (`out_ti` et `text`, respectivement). Enfin, vous créez la liaison entre les deux points de terminaison lorsque vous appelez le constructeur de la classe Binding (`new Binding(src, dest)`).

Il n'est pas nécessaire d'indiquer les noms de classes pleinement qualifiés (tels que `mx.data.binding.EndPoint`) dans votre code ActionScript, comme nous l'avons vu dans le premier exemple de code. Si vous utilisez l'instruction `import` au début de votre code, vous pouvez éviter l'emploi des noms pleinement qualifiés. Lorsque vous importez toutes les classes dans le package `mx.data.binding` via `(*)` (le package comprend les deux classes EndPoint et Binding), vous pouvez raccourcir votre code et référencer directement les classes EndPoint et Binding. Pour plus d'informations sur les instructions d'importation, consultez l'entrée `import` dans *Guide de référence du langage ActionScript 2.0*.

8. Choisissez Contrôle > Tester l'animation pour tester le code dans l'environnement de test. Saisissez du texte dans le champ `in_ti` d.

   Dès que l'occurrence `in_ti` perd le focus (cliquez sur la scène et appuyez sur la touche de tabulation ou cliquez dans le second champ), Flash copie le texte que vous avez saisi dans `in_ti` dans le champ `out_ti`.

9. Choisissez Fichier > Enregistrer pour enregistrer les modifications.

Si vous souhaitez modifier le texte du champ `out_ti` saisi à l'exercice précédent, votre code peut devenir bien plus complexe. Si vous définissez les liaisons à l'aide de panneau Inspecteur de composants, vous créez par défaut une connexion bidirectionnelle. Cela signifie que si vous modifiez l'un des champs de texte sur la scène, l'autre change également. Lorsque vous créez des liaisons à l'aide d'ActionScript, votre application fonctionne à l'inverse. Les liaisons de données à l'exécution sont unidirectionnelles par défaut, sauf si vous spécifiez le contraire, comme dans l'exemple suivant.

Pour créer une liaison bidirectionnelle à l'aide de code ActionScript, quelques modifications mineures doivent être apportées aux fragments de code de la procédure précédente.
Cet exemple utilise le second code ActionScript abrégé du fragment de code de l'étape 7.

Pour créer une liaison bidirectionnelle :

1. Ouvrez le fichier **panel_as.fla** de l'exemple précédent.
2. Modifiez légèrement votre code ActionScript (voir le code en **gras**) pour qu'il corresponde au code ActionScript suivant :

```
import mx.data.binding.*;
var src:EndPoint = new EndPoint();
src.component = in_ti;
src.property = "text";
src.event = "focusOut";
var dest:EndPoint = new EndPoint();
dest.component = out_ti;
dest.property = "text";
dest.event = "focusOut";
new Binding(src, dest, null, true);
```

   Les deux modifications du code ActionScript effectuent les opérations suivantes :

   - Définition d'une propriété d'événement pour l'occurrence EndPoint de destination.
   - Définition de deux paramètres supplémentaires pour le constructeur Binding.

   Le premier paramètre est utilisé pour des options de mise en forme avancée. Cette valeur peut être définie sur `null` ou `undefined`. Le second paramètre indique si la liaison est bidirectionnelle (`true`) ou unidirectionnelle (`false`).

   Vous vous demandez peut-être d'où vient l'événement `focusOut`. C'est là que le code ActionScript se complique quelque peu. Vous pouvez examiner la classe TextInput et utiliser certaines des méthodes énumérées (telles que `change()` ou `enter()`), mais vous ne trouverez là aucun événement `focusOut`. La classe TextInput hérite des classes UIObject et UIComponent. Si vous examinez la classe UIComponent, qui ajoute la prise en charge du focus aux composants, vous y voyez quatre événements supplémentaires : `focusIn`, `focusOut`, `keyDown` et `keyUp`. Ces événements peuvent être utilisés avec le composant TextInput.

3. (Facultatif) Si vous souhaitez que l'exemple précédent actualise la valeur du champ `out_ti`, vous pouvez remplacer l'événement `focusOut` par `change`.
4. Sélectionnez Contrôle > Tester l'animation pour tester le document.

   Flash modifie la seconde valeur du champ `in_ti` et actualise celle de `out_ti`. Vous avez bien créé une connexion bidirectionnelle.

Vous pouvez utiliser les classes Liaison avec la plupart des composants de l'interface utilisateur de la version 2 de l'architecture des composants Adobe, et non pas uniquement avec le composant TextInput. L'exemple suivant montre comment utiliser ActionScript pour lier des occurrences CheckBox et des composants Label pendant l'exécution.

Pour utiliser des classes de liaison avec le composant CheckBox :

1. Créez un nouveau document Flash.
2. Sélectionnez le Fichier > Enregistrer sous et nommez le fichier **checkbox_as.fla**.
3. Sélectionnez la fenêtre > Bibliothèques communes> Classes.
4. Faites glisser une copie de la classe DataBindingClasses dans la bibliothèque du document.
5. Faites glisser une copie du composant CheckBox sur la scène et donnez-lui le nom d'occurrence **my_ch**.
6. Faites glisser une copie du composant Label sur la scène et donnez-lui le nom d'occurrence **my_lbl**.
7. Créez un nouveau calque et nommez-le **actions**.
8. Ajoutez le code ActionScript suivant dans l'image 1 du calque actions :

```
var srcEndPoint:Object = {component:my_ch, property:"selected",
  event:"click"};
var destEndPoint:Object = {component:my_lbl, property:"text"};
new mx.data.binding.Binding(srcEndPoint, destEndPoint);
```

   Au lieu de créer de nouvelles occurrences de la classe EndPoint, comme dans les exercices précédents de cette section, utilisez des objets pour définir les points de terminaison. Le code de cette étape crée deux objets jouant le rôle de points de terminaison pour la liaison. Vous créez la liaison lorsque vous appelez le constructeur de la classe Binding. Pour réduire encore davantage la quantité de code (et sa lisibilité), définissez les objets en ligne, comme dans l'exemple de fragment de code suivant :

```
new mx.data.binding.Binding({component:my_ch, property:"selected",
  event:"click"}, {component:my_lbl, property:"text"});
```

   Ce code ActionScript réduit la lisibilité de votre code, mais également la quantité de saisie nécessaire. Si vous partagez vos fichiers FLA (ou ActionScript), il est peut-être préférable d'utiliser le premier extrait de code ActionScript, car il est bien plus clair.

Utilisation de composants, de liaisons et de mises en forme personnalisées

Les mises en forme personnalisées simplifient le formatage des données complexes. Vous pouvez également les utiliser pour afficher des images, du texte HTML formaté ou d'autres composants dans un composant tel que DataGrid. L'exemple suivant montre l'utilité des mises en forme personnalisées.

Pour utiliser des mises en forme personnalisées dans un document :

1. Créez un nouveau fichier FLA et ajoutez la classe DataBindingClasses à la bibliothèque (Fenêtre > Bibliothèques communes > Classes).
2. Faites glisser une copie du composant DateChooser sur la scène et donnez-lui le nom d'occurrence **my_dc**.
3. Faites glisser une copie du composant Label sur la scène et donnez-lui le nom d'occurrence **my_lbl**.
4. Insérez un nouveau calque et nommez-le **actions**.
5. Dans l'Image 1 du calque actions, ajoutez le code ActionScript suivant :

```
import mx.data.binding.*;
var src:EndPoint = new EndPoint();
src.component = my_dc;
src.property = "selectedDate";
src.event = "change";
var dest:EndPoint = new EndPoint();
dest.component = my_lbl;
dest.property = "text";
new Binding(src, dest);
```

   Ce code crée une liaison entre la propriété `selectedDate` du composant DateChooser et la propriété `text` du composant Label sur la scène. Chaque fois que vous cliquez sur une nouvelle date du calendrier, celle-ci apparaît dans le composant Label.

6. Enregistrez le document Flash sous le nom **customformat.fla**, à l'emplacement de votre choix sur votre disque dur.

   (Vous le recyclerez dans l'exercice suivant.)

7. Sélectionnez Contrôle > Tester l'animation pour tester le document.

   Tentez de modifier les dates dans le composant Calendar. Vous verrez alors la date sélectionnée s'afficher dans le composant Label. Ce composant n'étant pas suffisamment large pour afficher la date complète, Flash perd une partie du texte.

8. Fermez le fichier SWF pour revenir dans l'environnement de programmation.

   Redimensionnez le composant Label sur la scène ou sélectionnez-le et définissez la propriété `autoSize` sur `left` dans l'onglet Paramètres de l'inspecteur Propriétés.

9. Choisissez Contrôle > Tester l'animation pour tester de nouveau le document.

   Le champ affiche maintenant la date complète, même si le manque de mise en forme est évident. Selon votre fuseau horaire et la date sélectionnée, la date peut ressembler à cela :

   ```
   Thu Nov 4 00:00:00 GMT-0800 2004
   ```

   Bien que la liaison fonctionne correctement et affiche la propriété `selectedDate`, ces dates ne sont pas très lisibles pour l'utilisateur. Personne ne souhaite connaître le décalage horaire, et il n'est peut-être pas nécessaire d'afficher les heures, les minutes et les secondes. Vous avez simplement besoin d'un format de date plus lisible et moins rébarbatif. Les mises en forme personnalisées se révèlent donc particulièrement utiles pour présenter le texte.

Mise en forme des données à l'aide de la classe 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.CustomFormatter`. La classe CustomFormatter vous permet de convertir des types de données en chaîne et inversement. Dans ce cas, il vous faut convertir la propriété `selectedDate` du composant DateChooser en une chaîne attrayante lorsque la valeur est copiée dans le composant Label.

L'exemple suivant décrit la création d'une mise en forme personnalisée affichant la date au format `NOV 4, 2004` au lieu de la chaîne de date par défaut.

REMARQUE

Avant de commencer cet exercice, achevez celui de la section « Utilisation de composants, de liaisons et de mises en forme personnalisées », à la page 581.

Pour mettre en forme des données à l'aide de la classe CustomFormatter :

1. Sélectionnez le fichier > Nouveau puis sélectionnez le fichier ActionScript pour créer un nouveau fichier AS.
2. Choisissez Fichier > Enregistrer sous et nommez le nouveau fichier **DateFormat.as**.

3. Saisissez le code suivant dans la fenêtre de script :

```
class DateFormat extends mx.data.binding.CustomFormatter {
  function format(rawValue:Date):String {
    var returnValue:String;
    var monthName_array:Array =
["JAN","FEB","MAR","APR","MAY","JUN","JUL","AUG","SEP","OCT","NOV","D
EC"];
    returnValue = monthName_array[rawValue.getMonth()]+"
"+rawValue.getDate()+", "+rawValue.getFullYear();
    return returnValue;
  }
}
```

La première section de code définit la nouvelle classe appelée DateFormat, qui étend la classe CustomFormatter dans le package `mx.data.binding`. N'oubliez pas que Flash compile les classes de liaison dans le fichier de composant DataBindingClasses et que, de ce fait, vous ne pouvez pas les voir directement ni les trouver dans le dossier Classes du répertoire d'installation de Flash.

La seule méthode que vous utilisez est la méthode `format()`, qui convertit l'occurrence de la date en un format de chaîne personnalisé. L'étape suivante consiste à créer un tableau des noms de mois de sorte que le résultat final ressemble à `NOV 4, 2004` plutôt qu'au format de date par défaut. N'oubliez pas que les tableaux étant basés sur zéro dans Flash, si la valeur de `rawValue.getMonth()` renvoie `1`, cela correspond au mois de février et pas au mois de janvier (ce dernier étant le mois `0`). Le code restant crée la chaîne de format personnalisé en concaténant les valeurs et en renvoyant la chaîne `returnValue`.

Vous risquez de rencontrer un problème en travaillant avec des classes figurant dans un clip compilé,ce que vous pouvez constater dans le fragment de code précédent. Comme vous étendez une classe qui est située dans la classe DataBindingClasses et non directement disponible pour Flash, l'erreur suivante survient lorsque vous vérifiez la syntaxe de la classe précédente :

```
**Error** <path to DateFormat class>\DateFormat.as: Line 1: The class
  'mx.data.binding.CustomFormatter' could not be loaded.
     class DateFormat extends mx.data.binding.CustomFormatter {

Total ActionScript Errors: 1  Reported Errors: 1
```

Votre code est probablement correct. Ce problème survient lorsque Flash ne peut pas localiser la classe, ce qui entraîne l'échec de la vérification de la syntaxe.

4. Enregistrez le fichier DateFormat.as.
5. Ouvrez le fichier customformat.fla de l'exercice de la section « Utilisation de composants, de liaisons et de mises en forme personnalisées ». Assurez-vous d'enregistrer ou de copier DateFormat.as dans le même répertoire que ce fichier.

6. Dans customformat.fla, modifiez le code ActionScript de l'image 1 du calque actions en fonction du code suivant :

```
import mx.data.binding.*;
var src:EndPoint = new EndPoint();
src.component = my_dc;
src.property = "selectedDate";
src.event = "change";
var dest:EndPoint = new EndPoint();
dest.component = my_lbl;
dest.property = "text";
new Binding(src, dest, {cls:mx.data.formatters.Custom,
  settings:{classname:"DateFormat", classname_class:DateFormat}});
```

   Cette fois, vous définissez un objet `customFormatter` qui indique à Flash que vous utilisez la nouvelle classe DateFormat pour mettre en forme le point de terminaison de la liaison.

7. Enregistrez les modifications du document, puis choisissez Contrôle > Tester l'animation pour tester votre code.

Ajout et liaison de composants sur la scène

L'un des plus gros avantages de l'utilisation des classes de liaison avec ActionScript est que vous pouvez créer des liaisons entre les composants que Flash ajoute à la scène au moment de l'exécution. Imaginez que vous créez votre propre classe personnalisée qui ajoute des champs de texte appropriés à la scène au moment de l'exécution, puis valide les données utiles et ajoute les liaisons nécessaires. Tant que les composants sont dans votre bibliothèque, vous pouvez les ajouter de façon dynamique et créer les liaisons à l'aide de deux lignes de code supplémentaires.

Pour ajouter, puis lier des composants sur la scène à l'aide d'ActionScript :

1. Créez un nouveau document Flash.
2. Faites glisser deux composants, un ComboBox et un Label, dans la bibliothèque du document.
3. Insérez un nouveau calque et nommez-le **actions**.

4. Dans l'Image 1 du calque actions, ajoutez le code suivant :

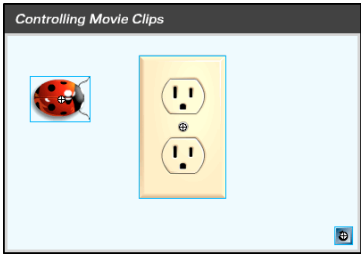
```
import mx.data.binding.*;
this.createClassObject(mx.controls.ComboBox, "my_cb", 1, {_x:10,
  _y:10});
this.createClassObject(mx.controls.Label, "my_lbl", 2, {_x:10, _y:40});
my_cb.addItem("JAN", 0);
my_cb.addItem("FEB", 1);
my_cb.addItem("MAR", 2);
my_cb.addItem("APR", 3);
my_cb.addItem("MAY", 4);
my_cb.addItem("JUN", 5);
var src:EndPoint = new EndPoint();
src.component = my_cb;
src.property = "value";
src.event = "change";
var dest:EndPoint = new EndPoint();
dest.component = my_lbl;
dest.property = "text";
new Binding(src, dest);
```

La première ligne de code ActionScript importe les classes du package `mx.data.binding` de sorte que les chemins complets ne sont plus nécessaires dans votre code. Les deux lignes de code ActionScript suivantes joignent les composants de la bibliothèque du document à la scène. Vous placez ensuite les composants sur la scène.

Enfin, vous ajoutez les données à l'occurrence ComboBox et créez la liaison entre les composants ComboBox `my_cb` et Label `my_lbl` sur la scène.

Structure d'un exemple de script

Lorsqu'un utilisateur fait glisser la coccinelle vers la prise électrique dans le fichier d'exemple zapper.swf (disponible dans le guide Utilisation de Flash de l'aide), la coccinelle tombe et la prise tremble. Le scénario principal ne comprend qu'une image et contient trois objets : la coccinelle , la prise et un bouton de réinitialisation. Chacun de ces objets est une occurrence de clip.



Le script suivant est associé à l'image 1 du scénario principal :

```
var initx:Number = bug_mc._x;
var inity:Number = bug_mc._y;
var zapped:Boolean = false;

reset_btn.onRelease = function() {
  zapped = false;
  bug_mc._x = initx;
  bug_mc._y = inity;
  bug_mc._alpha = 100;
  bug_mc._rotation = 0;
};

bug_mc.onPress = function() {
  this.startDrag();
};
bug_mc.onRelease = function() {
  this.stopDrag();
};
bug_mc.onEnterFrame = function() {
  if (this.hitTest(this._parent.zapper_mc)) {
    this.stopDrag();
    zapped = true;
    bug_mc._alpha = 75;
    bug_mc._rotation = 20;
    this._parent.zapper_mc.play();
  }
  if (zapped) {
    bug_mc._y += 25;
  }
};
```

L'occurrence de coccinelle s'appelle `bug_mc` et l'occurrence de prise est `zapper_mc`. Dans le script, la référence à la coccinelle est `this`, car le script est associé à la coccinelle et le mot réservé `this` désigne l'objet qui le contient.

Des gestionnaires d'événement sont associés à plusieurs événements : `onRelease()`, `onPress()` et `onEnterFrame()`. Les gestionnaires d'événement sont définis sur l'image 1 après le chargement du fichier SWF. Les actions contenues dans le gestionnaire d'événement `onEnterFrame()` s'exécutent à chaque fois que la tête de lecture entre dans une image. Même dans un fichier SWF composé d'une image unique, la tête de lecture entre plusieurs fois dans cette image et le script est exécuté à plusieurs reprises.

Deux variables, `initx` et `inity`, sont définies de manière à stocker les coordonnées *x* et *y* initiales de l'occurrence de clip `bug_mc`. Une fonction est définie et affectée au gestionnaire d'événement `onRelease` de l'occurrence `reset_btn`. Cette fonction est appelée à chaque clic de la souris sur le bouton de réinitialisation, `reset_btn`. La fonction replace la coccinelle en position de départ sur la scène, réinitialise ses valeurs alpha et de rotation et redéfinit la variable `zapped` sur la valeur `false`.

Une instruction conditionnelle `if` utilise la méthode `hitTest` pour vérifier si l'occurrence de la coccinelle touche l'occurrence de la prise (`this._parent.zapper_mc`). Cette évaluation peut avoir deux types de résultats, `true` ou `false` :

- Si la méthode `hitTest()` renvoie la valeur `true`, Flash appelle la méthode `stopDrag()`, définit la variable `zapped` sur `true`, modifie les propriétés alpha et rotation et lance l'exécution de l'occurrence `zapper_mc`.
- Si la méthode `hitTest()` renvoie la valeur `false`, le code entre `{}`, qui suit immédiatement l'instruction `if`, n'est pas exécuté.

Les actions de l'instruction `onPress()` sont exécutées une fois le bouton de la souris appuyé au-dessus de l'occurrence `bug_mc`. Les actions de l'instruction `onRelease()` sont exécutées une fois le bouton de la souris relâché au-dessus de l'occurrence `bug_mc`.

L'action `startDrag()` permet de faire glisser la coccinelle. Le script étant associé à l'occurrence `bug_mc`, le mot-clé `this` indique que c'est l'occurrence `bug` qui peut être déplacée :

```
bug_mc.onPress = function() {
  this.startDrag();
};
```

L'action `stopDrag()` interrompt le mouvement :

```
bug_mc.onRelease = function() {
  this.stopDrag();
};
```

CHAPITRE 14

Utilisation des images, du son et de la vidéo

14

Si vous importez une image ou un son pendant la création d'un document dans Flash CS3 Professional, cet élément est préparé et stocké dans le fichier SWF lorsque vous le publiez. Vous pouvez importer non seulement des fichiers de médias pendant la programmation mais aussi des fichiers de médias externes, notamment au format SWF, pendant l'exécution. Plusieurs raisons peuvent justifier de conserver les fichiers de supports en dehors d'un document Flash.

Réduire la taille des fichiers En conservant les fichiers média volumineux en dehors de votre document Flash et en les chargeant lors de l'exécution, vous pouvez réduire le délai initial de téléchargement de vos applications et présentations, particulièrement dans le cas de connexions Internet lentes.

Moduler les présentations volumineuses Vous pouvez diviser une présentation ou une application volumineuse en fichiers SWF séparés, puis les charger au fur et à mesure à l'exécution. Ce processus permet non seulement de réduire le délai initial de téléchargement, mais également de conserver et de mettre à jour le contenu de la présentation plus facilement.

Séparer le contenu de la présentation Cette technique est très commune dans le développement d'applications, notamment d'applications orientées données. Par exemple, une application de panier d'achat en ligne peut afficher une image de chaque produit. En chargeant chaque image à l'exécution, vous pouvez facilement mettre à jour l'image d'un produit sans modifier le fichier FLA d'origine.

Bénéficier des fonctions d'exécution seule Certaines fonctions, telles que le chargement dynamique des fichiers vidéo Flash (FLV) et MP3, ne sont accessibles d'ActionScript que pendant l'exécution.

Cette section explique comment utiliser des fichiers d'images, des fichiers de son et la vidéo FLV dans vos applications Flash. Pour plus d'informations, consultez les sections suivantes :

Chargement et utilisation de fichiers de médias externes

Vous pouvez charger plusieurs types de fichiers de médias dans une application Flash pendant l'exécution : les formats SWF, MP3, JPEG, GIF, PNG et FLV. Toutefois, toutes les versions de Flash Player ne prennent pas en charge tous les formats de médias. Pour plus d'informations sur les types de fichiers d'images pris en charge dans Flash Player 8 et les versions ultérieures, consultez la section « Chargement de fichiers SWF et de fichiers d'images externes », à la page 591. Pour plus d'informations sur la prise en charge de la vidéo FLV dans Flash Player, consultez la section « Utilisation du format vidéo FLV », à la page 601.

Flash Player peut charger un média externe depuis n'importe quelle adresse HTTP ou FTP, un disque local utilisant un chemin relatif ou à l'aide du protocole `file://`.

Pour charger des fichiers d'images et des fichiers SWF externes, vous pouvez utiliser la fonction `loadMovie()` ou `loadMovieNum()` , ou bien la méthode `MovieClip.loadMovie()` ou `MovieClipLoader.loadClip()`. Les méthodes de classe offrent généralement davantage de fonctions et de souplesse que les fonctions globales et s'adaptent bien aux applications complexes. Lorsque vous chargez un fichier SWF ou un fichier d'image, vous spécifiez un clip ou un fichier SWF comme cible de ce média. Pour plus d'informations sur le chargement des fichiers SWF et image, consultez la section « Chargement de fichiers SWF et de fichiers d'images externes », à la page 591.

Pour lire un fichier MP3 externe, utilisez la méthode `loadSound()` de la classe Sound. Cette méthode vous permet de spécifier si le fichier MP3 doit se télécharger progressivement ou être totalement téléchargé avant de démarrer la lecture. Vous pouvez également lire les informations ID3 intégrées dans les fichiers MP3, si elles sont disponibles. Pour plus d'informations, voir « Lecture des balises ID3 dans les fichiers MP3 », à la page 599.

Flash Video est le format vidéo natif utilisé par Flash Player. Vous pouvez lire les fichiers FLV sur HTTP ou sur un système de fichiers local. La lecture des fichiers FLV externes offre plusieurs avantages par rapport à l'intégration de la vidéo dans un document Flash : une amélioration des performances et de la gestion de la mémoire, et une cadence vidéo indépendante de la cadence Flash. Pour plus d'informations, voir « Lecture dynamique des fichiers FLV externes », à la page 604.

Vous pouvez également précharger un média externe ou suivre la progression de son téléchargement à l'aide de la classe MovieClipLoader, qui vous permet de suivre la progression du téléchargement des fichiers SWF ou d'images. Pour précharger des fichiers MP3 FLV, vous pouvez utiliser la méthode `getBytesLoaded()` de la classe Sound et la propriété `bytesLoaded` de la classe NetStream. Pour plus d'informations, voir « Préchargement de fichiers FLV », à la page 608.

Pour des exemples d'applications de galerie de photos, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Galleries afin d'accéder à ces exemples.

- gallery_tree.fla
- gallery_tween.fla

Ces fichiers vous montrent comment utiliser le code ActionScript pour contrôler dynamiquement des clips tout en chargeant des fichiers image dans un fichier SWF.

Chargement de fichiers SWF et de fichiers d'images externes

Pour charger un fichier SWF ou un fichier d'image, utilisez la fonction globale `loadMovie()` ou `loadMovieNum()`, ou la méthode `loadMovie()` de la classe MovieClip ou la méthode `loadClip()` de la classe MovieClipLoader. Pur plus d'informations sur la méthode `loadClip()`, consultez `MovieClipLoader.loadClip()` dans le *Guide de référence du langage ActionScript 2.0*.

Pour les fichiers d'images, Flash Player 8 et les versions ultérieures prennent en charge le format JPEG (progressif et non progressif), les images GIF (transparentes et non transparentes, bien que seule la première image d'un GIF animé soit chargée) et les fichiers PNG (transparents et non transparents).

Pour charger un fichier SWF ou un fichier d'image dans un niveau de Flash Player, utilisez la fonction `loadMovieNum()`. Pour charger un fichier SWF ou un fichier d'image dans une cible de clip, utilisez la fonction ou méthode `loadMovie()`. Dans les autres cas, le contenu chargé remplace le contenu du niveau ou du clip cible spécifié.

Lorsque vous chargez un fichier SWF ou un fichier d'image dans un clip cible, le coin supérieur gauche du fichier SWF ou du fichier d'image est placé sur le point d'alignement du clip. Puisque ce point se trouve souvent au centre du clip, il se peut que le contenu chargé ne soit pas centré. De même, lorsque vous chargez un fichier SWF ou une image dans un scénario racine, son coin supérieur gauche est placé sur le coin supérieur gauche de la scène. Le contenu chargé hérite de la rotation et de la mise à l'échelle du clip, mais le contenu d'origine du clip est supprimé.

Vous pouvez éventuellement envoyer des variables ActionScript en appelant la fonction `loadMovie()` ou `loadMovieNum()`. Cette technique est particulièrement utile si, par exemple, l'URL spécifiée dans l'appel d'une méthode est un script côté serveur qui renvoie un fichier SWF ou un fichier d'image en fonction des données transmises par l'application Flash.

Lorsque vous utilisez la fonction globale `loadMovie()` ou `loadMovieNum()`, spécifiez le niveau ou le clip cible en tant que paramètre. L'exemple suivant permet de charger le fichier d'animation Flash contents.swf dans l'occurrence de clip appelée `image_mc` :

```
loadMovie("contents.swf", image_mc);
```

Vous pouvez utiliser `Clip.loadMovie()` pour obtenir le même résultat :

```
image_mc.loadMovie("contents.swf");
```

Le code suivant charge l'image JPEG image1.jpg dans l'occurrence de clip `image_mc` :

```
image_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
```

Pour plus d'iinformations sur le chargement des fichiers SWF et des fichiers d'images, consultez la section « A propos du téléchargement des fichiers SWF et du scénario racine », à la page 595.

Pour précharger les fichiers SWF et JPEG dans les occurrences de clip, vous pouvez utiliser la classe MovieClipLoader. Cette classe offre un mécanisme d'écouteur d'événements qui indique l'état des téléchargements de fichiers dans les clips. Pour utiliser un objet MovieClipLoader pour précharger des fichiers SWF et JPEG, vous devez suivre les étapes ci-dessous :

Créer un objet MovieClipLoader Vous pouvez utiliser un seul objet MovieClipLoader pour suivre la progression du téléchargement de plusieurs fichiers ou créer un objet distinct pour suivre la progression de chaque fichier. Créez un clip, chargez le contenu dans ce dernier, puis créez l'objet MovieClipLoader comme illustré dans le code suivant :

```
this.createEmptyMovieClip("img_mc", 999);
var my_mcl:MovieClipLoader = new MovieClipLoader();
```

Créer un objet écouteur et créer des gestionnaires d'événement L'objet écouteur peut être tout objet ActionScript, par exemple un objet Object générique, un clip ou un composant personnalisé.

L'exemple suivant crée écouteur générique appelé `loadListener` et définit pour lui-même les fonctions `onLoadError`, `onLoadStart`, `onLoadProgress` et `onLoadComplete`.

```
// Crée un objet écouteur :
var mclListener:Object = new Object();
mclListener.onLoadError = function(target_mc:MovieClip, errorCode:String,
  status:Number) {
  trace("Error loading image: " + errorCode + " [" + status + "]");
};
mclListener.onLoadStart = function(target_mc:MovieClip):Void {
  trace("onLoadStart: " + target_mc);
};
mclListener.onLoadProgress = function(target_mc:MovieClip,
  numBytesLoaded:Number, numBytesTotal:Number):Void {
  var numPercentLoaded:Number = numBytesLoaded / numBytesTotal * 100;
```

```
  trace("onLoadProgress: " + target_mc + " is " + numPercentLoaded + "%
  loaded");
};
mclListener.onLoadComplete = function(target_mc:MovieClip,
  status:Number):Void {
  trace("onLoadComplete: " + target_mc);
};
```

REMARQUE

Flash Player 8 et les versions ultérieures permettent de vérifier le statut HTTP d'un téléchargement MovieClipLoader dans les écouteurs d'événement onLoadComplete et onLoadError. Il est ainsi possible de savoir pourquoi le fichier n'a pas pu être téléchargé (que ce soit à cause d'une erreur de serveur, d'un fichier introuvable, etc.).

Enregistrer l'objet d'écoute avec l'objet MovieClipLoader Vous devez enregistrer l'objet d'écoute avec l'objet MovieClipLoader pour qu'il reçoive les événements de chargement, comme dans l'exemple de code suivant :

```
my_mcl.addListener(mclListener);
```

Commencer à charger le fichier (image ou SWF) dans un clip cible Pour commencer le téléchargement du fichier d'image ou du fichier SWF, utilisez la méthode `MovieClipLoader.loadClip()`, comme dans l'exemple de code suivant :

```
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);
```

REMARQUE

Vous ne pouvez utiliser les méthodes MovieClipLoader que pour suivre la progression du téléchargement des fichiers chargés avec la méthode `MovieClipLoader.loadClip()`. Vous ne pouvez pas utiliser la fonction `loadMovie()` ou la méthode `MovieClip.loadMovie()`.

L'exemple suivant utilise la méthode `setProgress()` du composant ProgressBar pour afficher la progression du téléchargement d'un fichier SWF. Pour plus d'informations, consultez le manuel *Référence du langage des composants ActionScript 2.0*.

Pour afficher la progression du téléchargement en utilisant le composant ProgressBar :

1. Créez un document Flash, puis enregistrez-le sous le nom de **progress.fla**.
2. Ouvrez le panneau Composants (Fenêtre > Composants).
3. Faites glisser un composant ProgressBar du panneau Composants à la scène.
4. Dans l'inspecteur des propriétés (Fenêtre > Propriétés > Propriétés), appelez le composant ProgressBar `my_pb`.
5. Sélectionnez l'image 1 dans le scénario et ouvrez le panneau Actions (Fenêtre > Actions).

6. Ajoutez le code suivant au panneau Actions :

```
var my_pb:mx.controls.ProgressBar;
my_pb.mode = "manual";

this.createEmptyMovieClip("img_mc", 999);

var my_mcl:MovieClipLoader = new MovieClipLoader();
var mclListener:Object = new Object();
mclListener.onLoadStart = function(target_mc:MovieClip):Void {
  my_pb.label = "loading: " + target_mc._name;
};
mclListener.onLoadProgress = function(target_mc:MovieClip,
  numBytesLoaded:Number, numBytesTotal:Number):Void {
  var pctLoaded:Number = Math.ceil(100 * (numBytesLoaded /
  numBytesTotal));
  my_pb.setProgress(numBytesLoaded, numBytesTotal);
};
my_mcl.addListener(mclListener);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);
```

7. Testez le document en choisissant Contrôle > Tester l'animation.

   L'image est chargée dans le clip `img_mc`.

8. Sélectionnez Fichier > Publier > Formats, et assurez-vous que les options SWF et HTML sont sélectionnées.

9. Cliquez sur Publier et recherchez les fichiers HTML et SWF sur votre disque dur.

   Ils sont dans le même dossier que le fichier progress.fla que vous avez enregistré à l'étape 1.

10. Double-cliquez sur le document HTML pour l'ouvrir dans un navigateur et voir la barre de progression s'animer.

REMARQUE	Lorsque vous chargez des fichiers dans l'environnement de test, assurez-vous de charger un fichier, qui n'est pas en mémoire cache, à partir d'Internet et non pas un fichier local pour visualiser la barre de progression. Les fichiers locaux se chargent trop rapidement pour que la barre de progression soit visible. Sinon, téléchargez votre fichier SWF et testez votre document sur un serveur.

Pour plus d'informations sur ce sujet, consultez la section « A propos du téléchargement des fichiers SWF et du scénario racine », à la page 595. Pour plus d'informations sur la classe MovieClipLoader, consultez `MovieClipLoader` dans le *Guide de référence du langage ActionScript 2.0*. Pour plus d'informations sur la création d'une animation de la barre de progression, consultez « Création d'une animation de progression pour le chargement de fichiers SWF et de fichiers d'images », à la page 624.

Pour des exemples d'applications de galerie de photos, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Galleries afin d'accéder à ces exemples.

- gallery_tree.fla
- gallery_tween.fla

Ces fichiers vous montrent comment utiliser le code ActionScript pour contrôler dynamiquement des clips tout en chargeant des fichiers image dans un fichier SWF.

A propos du téléchargement des fichiers SWF et du scénario racine

La propriété ActionScript `_root`, spécifie ou renvoie une référence au scénario racine d'un fichier SWF. Si vous chargez un fichier SWF dans le clip d'un autre fichier SWF, les références à `_root` dans le fichier SWF chargé se traduisent dans le scénario racine du fichier SWF hôte et non pas dans celui du fichier SWF chargé. Cette action peut parfois engendrer un comportement inattendu à l'exécution, par exemple, si le fichier SWF hôte et le fichier SWF chargé utilisent tous les deux `_root` pour spécifier une variable.

A partir de Flash Player 7, vous pouvez utiliser la propriété `_lockroot (propriété MovieClip._lockroot)` pour obliger les références à `_root` effectuées par un clip à se traduire dans son propre scénario, plutôt que dans celui de l'animation contenant le clip. Pour plus d'informations, voir « Spécification d'un scénario racine pour les fichiers SWF chargés », à la page 340. Pour plus d'informations sur l'utilisation de `_root` et `_lockroot`, consultez le Chapitre 17, « Recommandations et conventions de programmation pour ActionScript 2.0 », à la page 715.

Un fichier SWF peut en charger un autre à partir de n'importe quel emplacement d'Internet. Cependant, pour qu'un fichier SWF ait accès aux données (variables, méthodes, etc.) définies dans un autre fichier SWF, il faut que les deux fichiers proviennent du même domaine. Dans Flash Player 7 et ses versions ultérieures, toute écriture de scripts inter-domaines est interdite sauf si le fichier SWF chargé en décide autrement en appelant `System.security.allowDomain()`.

Pour plus d'informations sur System.security.allowDomain, consultez l'entrée `allowDomain (méthode security.allowDomain)` dans le *Guide de référence du langage ActionScript 2.0*, ainsi que la section « Restriction des API de réseau », à la page 695.

Chargement et utilisation de fichiers MP3 externes

Pour charger des fichiers MP3 à l'exécution, utilisez la méthode `loadSound()` de la classe Sound. Tout d'abord, créez un objet Sound, comme indiqué dans l'exemple suivant :

```
var song1_sound:Sound = new Sound();
```

Utilisez ce nouvel objet pour appeler la méthode `loadSound()` et charger un événement ou un son lu en flux continu. Alors que les sons d'événement sont entièrement chargés avant leur lecture, les sons en flux continu sont lus pendant leur téléchargement. Vous pouvez définir le paramètre `isStreaming` de la méthode `loadSound()` de manière à définir un son comme étant un son lu en flux continu ou un son d'événement. Après avoir chargé un son d'événement, vous devez appeler la méthode `start()` de la classe Sound pour lancer sa lecture. La lecture des sons en flux continu débute dès qu'une quantité suffisante de données a été chargée dans le fichier SWF et la méthode `start()` n'est pas nécessaire.

Par exemple, le code suivant crée un objet Sound appelé `my_sound` et charge ensuite un fichier MP3 appelé song1.mp3. Entrez le code ActionScript suivant dans l'image 1 du scénario :

```
var my_sound:Sound = new Sound();
my_sound.loadSound("http://www.helpexamples.com/flash/sound/song1.mp3",
  true);
```

Dans la plupart des cas, définissez le paramètre `isStreaming` sur la valeur `true`, particulièrement si vous chargez des fichiers audio volumineux dont la lecture doit démarrer dès que possible, par exemple, dans une application de type « juke-box » MP3. Cependant, si vous téléchargez des clips audio légers et devez les lire à un moment précis (par exemple, lorsqu'un utilisateur clique sur un bouton), définissez le paramètre `isStreaming` sur la valeur `false`.

Pour déterminer la fin du téléchargement d'un son, utilisez le gestionnaire d'événements `Son.onLoad`. Ce gestionnaire d'événements reçoit automatiquement une valeur booléenne (`true` ou `false`) qui indique si le fichier a été téléchargé correctement.

Pour plus d'informations, se reporter aux sections suivantes :

- « Chargement d'un fichier MP3 », à la page 597
- « Préchargement de fichiers MP3 », à la page 597
- « Lecture des balises ID3 dans les fichiers MP3 », à la page 599

Pour un exemple de fichier source, jukebox.fla, qui charge des fichiers MP3, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Jukebox afin d'accéder à cet exemple. Cet exemple montre comment créer un juke-box en utilisant les types de données, les principes de codage général, et plusieurs composants.

Chargement d'un fichier MP3

Imaginons que vous créez un jeu en ligne qui utilise différents sons en fonction du niveau atteint par l'utilisateur dans le jeu. Le code suivant charge un fichier MP3 (song2.mp3) dans l'objet Sound `game_sound`, puis le lit à la fin de son téléchargement.

Pour charger un fichier MP3 :

1. Créez un fichier FLA appelé **loadMP3.fla**.
2. Sélectionnez l'image 1 du scénario, puis tapez le code suivant dans le panneau Actions :

```
var game_sound:Sound = new Sound();
game_sound.onLoad = function(success:Boolean):Void {
  if (success) {
    trace("Sound Loaded");
    game_sound.start();
  }
};
game_sound.loadSound("http://www.helpexamples.com/flash/sound/
  song2.mp3", false);
```

3. Choisissez Contrôle > Tester l'animation pour tester le son.

Pour les fichiers audio, Flash Player ne prend en charge que le format MP3 à l'exécution.

Pour plus d'informations, consultez les entrées `Sound.loadSound()`, `Sound.start()`, et `Sound.onLoad` dans le *Guide de référence du langage ActionScript 2.0*. Pour plus d'informations sur le préchargement de fichiers MP3, consultez la section « Préchargement de fichiers MP3 », à la page 597. Pour plus d'informations sur la création d'une animation de la barre de progression lorsque vous chargez un fichier audio, consultez la section « Création d'une barre de progression pour le téléchargement de fichiers MP3 avec ActionScript », à la page 626.

Pour un exemple de fichier source, jukebox.fla, qui charge des fichiers MP3, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Jukebox afin d'accéder à cet exemple. Cet exemple montre comment créer un juke-box en utilisant les types de données, les principes de codage général, et plusieurs composants.

Préchargement de fichiers MP3

Pour précharger des fichiers MP3 vous pouvez utiliser la fonction `setInterval()` afin de créer un *mécanisme d'interrogation* qui vérifie les octets chargés par un objet Sound ou NetStream à intervalles prédéfinis. Pour suivre la progression du téléchargement des fichiers MP3, utilisez les méthodes `Sound.getBytesLoaded()` et `Sound.getBytesTotal()`.

L'exemple suivant utilise `setInterval()` pour vérifier les octets chargés par un objet Sound à des intervalles prédéterminés.

Pour précharger un fichier MP3 :

1. Créez un fichier FLA appelé **preloadMP3.fla**.
2. Sélectionnez l'image 1 du scénario, puis tapez le code suivant dans le panneau Actions :

```
// Créez un nouvel objet Sound pour lire le son.
var songTrack:Sound = new Sound();
// Crée la fonction d'interrogation qui suit la progression du
// téléchargement.
// Il s'agit de la fonction qui est « interrogée ». Elle vérifie
// la progression du téléchargement de l'objet Sound transmis
// comme référence.
function checkProgress (soundObj:Object):Void {
  var numBytesLoaded:Number = soundObj.getBytesLoaded();
  var numBytesTotal:Number = soundObj.getBytesTotal();
  var numPercentLoaded:Number = Math.floor(numBytesLoaded /
  numBytesTotal * 100);
  if (!isNaN(numPercentLoaded)) {
    trace(numPercentLoaded + "% loaded.");
  }
};
// Lorsque le chargement du fichier est terminé, supprimez l'intervalle
// d'interrogation.
songTrack.onLoad = function ():Void {
  trace("load complete");
  clearInterval(poll);
};
// Charge le fichier MP3 en flux continu et commence à appeler
// checkProgress().
songTrack.loadSound("http://www.helpexamples.com/flash/sound/song1.mp3",
  true);
var poll:Number = setInterval(checkProgress, 100, songTrack);
```

3. Choisissez Contrôle > Tester l'animation pour tester le son.

   Le panneau de sortie affiche la progression du chargement.

Vous pouvez utiliser la technique d'interrogation pour précharger des fichiers FLV externes. Pour obtenir le nombre total d'octets et le nombre d'octets chargés pour un fichier FLV, utilisez les propriétés `NetStream.bytesLoaded` et `NetStream.bytesTotal` (pour plus d'informations, consultez les entrées bytesLoaded (propriété NetStream.bytesLoaded) et bytesTotal (propriété NetStream.bytesTotal)).

Pour plus d'informations, consultez les entrées `MovieClip.getBytesLoaded()`, `MovieClip.getBytesTotal()`, `setInterval()`, `Sound.getBytesLoaded()`, et `Sound.getBytesTotal()` dans le *Guide de référence du langage ActionScript 2.0*.

Pour plus d'informations sur la création d'une animation de la barre de progression, consultez « Création d'une barre de progression pour le téléchargement de fichiers MP3 avec ActionScript », à la page 626.

Pour un exemple de fichier source, jukebox.fla, qui charge des fichiers MP3, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Jukebox afin d'accéder à cet exemple. Cet exemple montre comment créer un juke-box en utilisant les types de données, les principes de codage général, et plusieurs composants.

Lecture des balises ID3 dans les fichiers MP3

Les balises ID3 sont des champs de données qui sont ajoutés à un fichier MP3. Les balises ID3 contiennent des informations sur le fichier, comme le nom du morceau, de l'album et de l'artiste.

Pour lire les balises ID3 d'un fichier MP3, utilisez la propriété `Sound.ID3`, dont les propriétés correspondent aux noms des balises ID3 incluses dans le fichier MP3 en cours de téléchargement. Pour déterminer la disponibilité des balises ID3 dans un fichier MP3 en cours de téléchargement, utilisez le gestionnaire d'événements`Sound.onID3`. Flash Player 7 prend en charge les balises des versions 1.0, 1.1, 2.3 et 2.4 ; les balises de la version 2.2 ne sont pas prises en charge.

L'exemple suivant charge un fichier MP3 appelé song1.mp3 dans l'objet Sound `song_sound`. Lorsque les balises ID3 sont disponibles pour le fichier, le champ de texte appelé `display_txt` affiche le nom de l'artiste et du morceau.

Pour lire les balises ID3 à partir d'un fichier MP3 :

1. Créez un fichier FLA appelé **id3.fla**.
2. Sélectionnez l'image 1 du scénario, puis tapez le code suivant dans le panneau Actions :

```
this.createTextField("display_txt", this.getNextHighestDepth(), 0, 0,
  100, 100);
display_txt.autoSize = "left";
display_txt.multiline = true;
var song_sound:Sound = new Sound();
song_sound.onLoad = function() {
  song_sound.start();
};
song_sound.onID3 = function():Void  {
  display_txt.text += "Artist:\t" + song_sound.id3.artist + "\n";
  display_txt.text += "Song:\t" + song_sound.id3.songname + "\n";
};
song_sound.loadSound("http://www.helpexamples.com/flash/sound/
  song1.mp3");
```

3. Choisissez Contrôle > Tester l'animation pour tester le son.

   Les balises ID3 apparaissent sur la scène, et le son est lu.

Les balises ID3 2.0 se trouvant au début d'un fichier MP3 (avant les données audio), elles sont disponibles dès que le fichier démarre le téléchargement. Cependant, comme les balises ID3 1.0 se trouvent à la fin du fichier (après les données audio), elles ne sont pas disponibles avant la fin du téléchargement du fichier MP3.

Le gestionnaire d'événements `onID3` est appelé à chaque fois que de nouvelles données ID3 sont disponibles. Cela signifie que si un fichier MP3 contient des balises ID3 2.0 et ID3 1.0, le gestionnaire d'événements `onID3` est appelé deux fois, car les balises se trouvent dans des parties différentes du fichier.

Pour obtenir la liste des balises ID3 prises en charge, consultez la section `id3 (propriété Sound.id3)` du *Guide de référence du langage ActionScript 2.0*.

Pour un exemple de fichier source, jukebox.fla, qui charge des fichiers MP3, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Jukebox afin d'accéder à cet exemple. Cet exemple montre comment créer un juke-box en utilisant les types de données, les principes de codage général et plusieurs composants.

Affectation de liaisons aux éléments de la bibliothèque.

Vous pouvez affecter des identifiants de liaison à des actifs de la bibliothèque, tels que des clips et des symboles de police. Dans Flash, vous pouvez définir les identifiants de liaison sur des éléments audio et des éléments d'image de la bibliothèque. Il est notamment possible d'utiliser des fichiers d'images et des fichiers audio avec des bibliothèques partagées et avec la nouvelle classe BitmapData.

L'exemple suivant ajoute une image bitmap à la bibliothèque avec une liaison définie sur `myImage`. Puis vous ajoutez une image sur la scène et la rendez déplaçable.

Pour utiliser la liaison avec des fichiers bitmap :

1. Créez un fichier FLA appelé **linkBitmap.fla**.
2. Importez une image bitmap dans la bibliothèque.
3. Cliquez avec le bouton droit (Windows) ou avec la touche Contrôle enfoncée (Macintosh) sur l'image dans la bibliothèque et choisissez Liaison dans le menu contextuel.
4. Sélectionnez Exporter pour ActionScript, puis Exporter dans la première image, et tapez **myImage** dans le champ Identifiant.
5. Cliquez sur OK pour définir l'identifiant de liaison.

6. Sélectionnez l'image 1 du scénario, puis tapez le code suivant dans le panneau Actions :

```
import flash.display.BitmapData;
// Crée imageBmp et attache le bitmap à partir de la bibliothèque
var imageBmp:BitmapData = BitmapData.loadBitmap("myImage");
// Crée un clip et attache imageBmp
this.createEmptyMovieClip("imageClip", 10);
imageClip.attachBitmap(imageBmp, 2);
// Rend le clip déplaçable
imageClip.onPress = function() {
     this.startDrag();
};
imageClip.onRelease = function() {
     this.stopDrag();
}
```

7. Sélectionnez Contrôle > Tester l'animation pour tester le document.

   L'image bitmap de la bibliothèque apparaît sur la scène, et l'image est déplaçable.

Utilisation du format vidéo FLV

Le format vidéo FLV contient des données audio et vidéo codées pour un acheminement via Flash Player. Par exemple, si vous avez un fichier vidéo QuickTime ou Windows Media, vous utilisez un encodeur (tel que Flash 8 Video Encoder ou Sorensen Squeeze) pour convertir ce fichier en fichier FLV.

Flash Player 7 prend en charge les fichiers FLV codés avec le codec vidéo Sorenson Spark. Flash Player 8 et les versions ultérieures prennent en charge les fichiers FLV codés avec l'encodeur Sorenson Spark ou On2 VP6 dans Flash. Le codec On2 VP6 prend en charge un canal alpha. Le format VLF est pris en charge différemment suivant la version de Flash. Pour plus d'informations, consultez le tableau suivant :

Codec	Version de fichier SWF (version publiée)	Version Flash Player requise pour la lecture
Sorenson Spark	6	6 et version ultérieure
	7	7 et version ultérieure
On2 VP6	6	8* et version ultérieure
	7	8 et version ultérieure
	8 et version ultérieure	8 et version ultérieure

* Si votre fichier SWF charge un fichier FLV, vous pouvez utiliser la vidéo On2 VP6 en ayant à publier de nouveau votre fichier SWF pour Flash Player 8 et versions ultérieures, tant que les utilisateurs exécutent Flash Player 8 et versions ultérieures pour afficher votre fichier SWF. Flash Player 8 et versions ultérieures prennent en charge la publication et la lecture des vidéos On2 VP6.

Pour plus de notions sur la vidéo, telles que le flux continu, le téléchargement progressif, les dimensions, le codage, l'importation et les griefs de bande passante, consultez le guide *Utilisation de Flash*.

Cette section traite de l'utilisation du format vidéo FLV sans composants. Vous pouvez aussi utiliser le composant FLVPlayback pour lire les fichiers FLV ou utiliser la classe VideoPlayback pour créer un lecteur vidéo personnalisé qui charge les fichiers FLV dynamiquement (consultez le site http://www.adobe.com/fr/devnet/flash/ ou http://www.adobe.com/support/documentation/fr/). Pour plus d'informations sur l'utilisation de la vidéo FLV avec les composants FLVPlayback et Media, consultez la section sur les composants Component et Media dans la *Référence du langage des composants ActionScript 2.0*.

Plutôt que d'importer directement des données vidéo dans l'environnement de programmation de Flash, vous pouvez utiliser ActionScript pour lire dynamiquement les fichiers FLV externes dans Flash Player. Vous pouvez lire les fichiers FLV à partir d'une adresse HTTP ou d'un système de fichiers local. Pour lire les fichiers FLV, utilisez les classes NetConnection et NetStream, ainsi que la méthode `attachVideo()` de la classe Video. Pour plus d'informations, consultez les entrées `NetConnection`, `NetStream` et `attachVideo (méthode Video.attachVideo)` du *Guide de référence du langage ActionScript 2.0*.

Vous pouvez créer des fichiers FLV en important la vidéo dans l'outil de programmation Flash et en l'exportant sous la forme d'un fichier FLV. Si vous disposez de Flash, vous pouvez utiliser le module d'exportation FLV pour exporter des fichiers FLV à partir des applications d'édition vidéo prises en charge.

L'utilisation des fichiers FLV externes offre certaines fonctionnalités qui ne sont pas disponibles avec l'utilisation de la vidéo importée :

- Les clips vidéo longs peuvent être utilisés dans les documents Flash sans ralentir la lecture. Les fichiers FLV externes sont lus à l'aide de la *mémoire cache*, ce qui signifie que les fichiers volumineux sont enregistrés en petites parties et sont accessibles dynamiquement. Ils nécessitent donc moins de mémoire que les fichiers vidéo intégrés.
- Un fichier FLV externe peut avoir une cadence différente de celle du document Flash dans lequel il est lu. Par exemple, vous pouvez définir la cadence du document Flash sur 30 ips (images par seconde) et celle de l'image vidéo sur 21 ips. Ce réglage vous offre un meilleur contrôle de la vidéo que la vidéo intégrée, pour assurer une lecture vidéo fluide. Il vous permet aussi de lire les fichiers FLV à différentes cadences d'images sans avoir à altérer un contenu Flash existant.

- Avec les fichiers FLV externes, la lecture du document Flash n'a pas besoin d'être interrompue pendant le chargement du fichier vidéo. Les fichiers vidéo importés peuvent parfois interrompre la lecture du document pour exécuter certaines fonctions, par exemple, accéder à un lecteur de CD-ROM. Les fichiers FLV peuvent exécuter les fonctions indépendamment du document Flash, ce qui n'interrompt pas la lecture.
- Le sous-titrage du contenu vidéo est plus facile avec les fichiers FLV externes, parce que vous utilisez les gestionnaires d'événements pour accéder aux métadonnées de la vidéo.

CONSEIL

Pour charger des fichiers FLV à partir d'un serveur Web, il peut être nécessaire d'enregistrer l'extension de fichier et le type MIME auprès de votre serveur Web. Pour cela, consultez la documentation du serveur. Le type MIME des fichiers FLV est video/x-flv. Pour plus d'informations, voir « Configuration de votre serveur pour les fichiers FLV », à la page 622.

Pour plus d'informations sur la vidéo FLV, consultez les rubriques suivantes :

Création d'un objet vidéo

Avant de charger et de manipuler la vidéo en utilisant ActionScript, vous devez créer un objet vidéo, le déplacer sur la scène, et lui donner un nom d'occurrence. L'exemple suivant explique comment ajouter une occurrence vidéo à une application.

Pour créer un objet vidéo :

1. Alors que votre document est ouvert dans l'outil de programmation Flash, sélectionnez Nouvelle vidéo dans le menu contextuel dans le panneau Bibliothèque (Fenêtre > Bibliothèque),
2. Dans la boîte de dialogue Propriétés vidéo, nommez le symbole vidéo et sélectionnez Vidéo (contrôlé par ActionScript).
3. Cliquez sur OK pour créer un objet vidéo.

4. Faites glisser l'objet vidéo du panneau Bibliothèque à la scène pour en créer une occurrence.
5. Lorsque l'objet vidéo est sélectionné sur la scène, dans l'inspecteur des propriétés (Fenêtre> Propriétés> Propriétés), saisissez **my_video** dans le champ de texte Nom de l'occurrence.

   Vous disposez maintenant d'une occurrence de la vidéo sur la scène, pour laquelle vous pouvez ajouter du code ActionScript afin de charger la vidéo ou de manipuler l'occurrence de diverses façons.

Pour plus d'informations sur le chargement de fichiers FLV dynamique, consultez la section « Lecture dynamique des fichiers FLV externes ». Pour plus d'informations sur la création d'un bandeau vidéo, consultez la section « Création d'un bandeau vidéo », à la page 605.

Lecture dynamique des fichiers FLV externes

Vous pouvez charger les fichiers FLV lors de l'exécution afin qu'ils soient lus dans un fichier SWF. Vous pouvez les charger dans un objet vidéo ou dans un composant tel que le composant FLVPlayback. L'exemple suivant montre comment lire un fichier appelé clouds.flv dans un objet vidéo

Pour lire un fichier FLV externe dans un document Flash :

1. Créez un document Flash appelé **playFLV.fla**.
2. Dans le panneau Bibliothèque (Fenêtre > Bibliothèque), choisissez Nouvelle vidéo à partir du menu déroulant Bibliothèque.
3. Dans la boîte de dialogue Propriétés vidéo, nommez le symbole vidéo et sélectionnez Vidéo (contrôlé par ActionScript).
4. Cliquez sur OK pour créer un objet vidéo.
5. Faites glisser l'objet vidéo du panneau Bibliothèque sur la scène pour créer une occurrence d'objet vidéo.
6. Lorsque l'objet vidéo est sélectionné sur la scène, dans l'inspecteur des propriétés (Fenêtre > Propriétés > Propriétés), saisissez **my_video** dans le champ de texte Nom de l'occurrence.
7. Sélectionnez l'image 1 dans le scénario et ouvrez le panneau Actions (Fenêtre > Actions).
8. Tapez le code suivant dans le panneau Actions :

```
this.createTextField("status_txt", 999, 0, 0, 100, 100);
status_txt.autoSize = "left";
status_txt.multiline = true;
// Crée un objet NetConnection
var my_nc:NetConnection = new NetConnection();
// Crée une connexion locale en flux continu
my_nc.connect(null);
```

```
// Crée un objet NetStream et définit une fonction onStatus()
var my_ns:NetStream = new NetStream(my_nc);
my_ns.onStatus = function(infoObject:Object):Void {
  status_txt.text += "status (" + this.time + " seconds)\n";
  status_txt.text += "\t Level: " + infoObject.level + "\n";
  status_txt.text += "\t Code: " + infoObject.code + "\n\n";
};
// Associe la source vidéo NetStream à l'objet Video
my_video.attachVideo(my_ns);
// Définit la durée du tampon
my_ns.setBufferTime(5);
// Lit le fichier FLV
my_ns.play("http://www.helpexamples.com/flash/video/clouds.flv");
```

9. Sélectionnez Contrôle > Tester l'animation pour tester le document.

Pour plus d'informations sur le préchargement de fichiers FLV, consultez la section « Préchargement de fichiers FLV », page 507. Pour plus d'informations sur le chargement dynamique de vidéo FLV dans des composants, consultez la *Référence du langage des composants*. Pour plus d'informations sur les fichiers FLV et le serveur ainsi que les fichiers FLV et la lecture des fichiers FLV localement sur le Macintosh, consultez la section « Configuration de votre serveur pour les fichiers FLV », à la page 622.

Création d'un bandeau vidéo

On utilise souvent du contenu vidéo au sein de bandeaux et autres publicités Flash, notamment pour diffuser des bandes annonces animées Flash ou des publicités de télévision. L'exemple suivant montre comment créer une occurrence vidéo et ajouter du code ActionScript dans un fichier FLA pour créer une publicité bandeau qui contient de la vidéo.

Pour créer un bandeau vidéo :

1. Créez un document Flash appelé **vidBanner.fla**.
2. Sélectionnez Modification > Document.
3. Modifiez les dimensions de votre fichier FLA, tapez **468** dans le champ largeur et **60** dans le champ hauteur.
4. Dans le panneau Bibliothèque (Fenêtre > Bibliothèque), choisissez Nouvelle vidéo à partir des options Bibliothèque.
5. Dans la boîte de dialogue Propriétés vidéo, nommez le symbole vidéo et sélectionnez Vidéo (contrôlé par ActionScript).
6. Cliquez sur OK pour créer un objet vidéo.
7. Faites glisser l'objet vidéo du panneau Bibliothèque à la scène pour créer une occurrence vidéo.

8. Lorsque l'objet vidéo est sélectionné sur la scène, dans l'inspecteur des propriétés (Fenêtre> Propriétés> Propriétés), saisissez **my_video** dans le champ de texte Nom de l'occurrence.
9. Avec l'occurrence vidéo encore sélectionnée, tapez **105** dans le champ largeur et **60** dans le champ hauteur dans l'inspecteur des propriétés.
10. Faites glisser l'occurrence video à l'endroit de votre choix de la scène, ou utilisez l'inspecteur des propriétés pour définir ses coordonnées *x* et *y*
11. Sélectionnez l'image 1 dans le scénario et ouvrez le panneau Actions (Fenêtre > Actions).
12. Ajoutez le code suivant au panneau Actions :

```
var my_nc:NetConnection = new NetConnection();
my_nc.connect(null);
var my_ns:NetStream = new NetStream(my_nc);
my_video.attachVideo(my_ns);
my_ns.setBufferTime(5);
my_ns.play("http://www.helpexamples.com/flash/video/vbanner.flv");
```

13. Choisissez Insertion > Scénario > Calque pour créer un calque et appelez-le **bouton**.
14. Choisissez l'outil Rectangle dans le panneau Outils.
15. Dans la section Couleurs du panneau Outils, cliquez sur l'icône de crayon pour sélectionner la commande Couleur de trait.
16. Choisissez Aucune couleur, ce qui supprime le contour du rectangle.
17. Cliquez et faites glisser le pointeur en diagonale sur la scène pour créer un rectangle.

    La taille du rectangle n'a pas d'importance parce que vous le redimensionnerez avec l'inspecteur des propriétés.
18. Choisissez l'outil de sélection dans le panneau Outil, puis cliquez sur le rectangle sur la scène pour le sélectionner.
19. Avec le rectangle encore sélectionné, tapez **468** dans le champ largeur et **60** dans le champ hauteur dans l'inspecteur des propriétés. Modifiez ensuite les coordonnées X et Y (zones de texte X et Y) de manière à ce qu'elles indiquent **0**.
20. Le rectangle sélectionné sur la scène. Appuyez sur F8 pour le convertir en symbole.
21. Dans la boîte de dialogue Convertir en symbole, saisissez **btn invisible** dans le champ Nom, choisissez Bouton, puis cliquez sur OK.

22. Double-cliquez sur le nouveau bouton sur la scène pour passer en mode d'édition de symbole.

    Le rectangle se trouve actuellement dans la première image Haut du bouton que vous avez créé. Il s'agit de l'état Haut du bouton – ce que les utilisateurs voient lorsque le bouton est sur la scène. Cependant, vous voulez que le bouton ne soit pas visible sur la scène. Vous devez donc déplacer le rectangle dans l'image Cliquable, qui représente la surface de réactivité du bouton (la région active sur laquelle un utilisateur peut cliquer afin d'activer les actions du bouton).

23. Cliquez sur l'image-clé au niveau de l'image Haut, et tout en maintenant enfoncé le bouton de la souris faites glisser l'image-clé vers l'image Cliquable.

    Vous pouvez maintenant cliquer sur toute la surface du bandeau, mais le bouton n'apparaît pas sur le bandeau.

24. Cliquez sur la Séquence 1 pour revenir au scénario principal.

    Un rectangle bleu sarcelle apparaît sur la surface du bandeau, représentant la surface Cliquable invisible du bouton.

25. Sélectionnez le bouton que vous avez créé, ouvrez l'inspecteur des propriétés, et saisissez **inv_btn** dans le champ Nom de l'occurrence.

26. Sélectionnez l'image 1 du scénario, puis tapez le code suivant dans le panneau Actions :

```
inv_btn.onRelease = function(){
  getURL("http://www.adobe.com");
};
```

27. Apportez d'autres modifications au bandeau. Vous pouvez par exemple ajouter des graphismes ou du texte.

28. Choisissez Contrôle > Tester l'animation pour tester le bandeau dans Flash Player.

Dans cet exemple , vous avez créé un bandeau et vous l'avez redimensionné sur la base des dimensions établies et normalisées que l'Interactive Advertising Bureau spécifie. Pour plus d'informations sur les dimensions standard (et pour prendre connaissance d'autres conseils bien utiles), consultez la page (en anglais) Standards and Guidelines du site Interactive Advertising Bureau à l'adresse www.iab.net/standards/adunits.asp.

Malgré les directives normalisées, n'oubliez pas de confirmer les règles à suivre auprès du service de publicité, de votre client ou du site web pour lequel vous travaillez pour la première fois. Si vous soumettez votre bandeau à une agence publicitaire, assurez-vous que le fichier correspond à leurs spécifications de taille, de dimensions, de version ciblée de Flash Player et de cadence. Par ailleurs, vous pouvez avoir à tenir compte d'autres informations, telles que les types d'éléments que vous pouvez utiliser, le code des boutons que vous employez dans le fichier FLA file, etc.

Préchargement de fichiers FLV

Pour suivre la progression du téléchargement des fichiers FLV, utilisez les propriétés `NetStream.bytesLoaded` et `NetStream.bytesTotal`. Pour obtenir le nombre total d'octets et le nombre d'octets chargés pour un fichier FLV, utilisez les propriétés `NetStream.bytesLoaded` et `NetStream.bytesTotal`.

L'exemple suivant utilise les propriétés `bytesLoaded` et `bytesTotal`, qui affichent la progression du chargement de video1.flv dans l'occurrence d'objet vidéo appelée my_video. Un champ texte appelé loaded_txt est créé de façon dynamique pour afficher des informations sur la progression du processus de chargement.

Pour précharger un fichier FLV :

1. Créez un fichier FLA appelé **preloadFLV.fla**.
2. Dans le panneau Bibliothèque (Fenêtre > Bibliothèque), choisissez Nouvelle vidéo dans le menu contextuel Bibliothèque.
3. Dans la boîte de dialogue Propriétés vidéo, nommez le symbole vidéo et sélectionnez Vidéo (contrôlé par ActionScript).
4. Cliquez sur OK pour créer un objet vidéo.
5. Faites glisser l'objet vidéo du panneau Bibliothèque sur la scène pour créer une occurrence d'objet vidéo.
6. Lorsque l'objet vidéo est sélectionné sur la scène, dans l'inspecteur des propriétés (Fenêtre> Propriétés> Propriétés), saisissez **my_video** dans le champ de texte Nom de l'occurrence.
7. Avec l'occurrence vidéo encore sélectionnée, tapez **320** dans le champ largeur et **213** dans le champ hauteur dans l'inspecteur des propriétés.
8. Sélectionnez l'image 1 dans le scénario et ouvrez le panneau Actions (Fenêtre > Actions).
9. Tapez le code suivant dans le panneau Actions :

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("http://www.helpexamples.com/flash/video/
  lights_short.flv");

this.createTextField("loaded_txt", this.getNextHighestDepth(), 10, 10,
  160, 22);
var loaded_interval:Number = setInterval(checkBytesLoaded, 500,
  stream_ns);
```

```
function checkBytesLoaded(my_ns:NetStream) {
  var pctLoaded:Number = Math.round(my_ns.bytesLoaded / my_ns.bytesTotal
  * 100);
  loaded_txt.text = Math.round(my_ns.bytesLoaded / 1000) + " of " +
  Math.round(my_ns.bytesTotal / 1000) + " KB loaded (" + pctLoaded +
  "%)";
  progressBar_mc.bar_mc._xscale = pctLoaded;
  if (pctLoaded >= 100) {
    clearInterval(loaded_interval);
  }
}
```

10. Choisissez Contrôle > Tester l'animation pour tester votre code.

REMARQUE	Si votre barre de progression se charge instantanément, la vidéo a été mise en cache sur votre disque dur (ou bien à partir du test de cet exemple ou de son chargement dans une procédure différente). Dans pareil cas, téléchargez un fichier FLV sur votre serveur et chargez-le à la place.

Pour précharger des fichiers FLV, vous pouvez aussi utiliser la méthode `NetStream.setBufferTime()`. Cette méthode admet un seul paramètre, qui indique le nombre de secondes du flux FLV à mettre en mémoire tampon avant le démarrage de la lecture. Pour plus d'informations, consultez les entrées `setBufferTime (méthode NetStream.setBufferTime)`, `getBytesLoaded (méthode MovieClip.getBytesLoaded)`, `getBytesTotal (méthode MovieClip.getBytesTotal)`, `bytesLoaded (propriété NetStream.bytesLoaded)`, `bytesTotal (propriété NetStream.bytesTotal)` et `fonction setInterval` dans le *Guide de référence du langage ActionScript 2.0.*

Utilisation des points de repère

Vous pouvez utiliser plusieurs types de points de repère avec Flash Video. Vous pouvez utiliser ActionScript pour interagir avec les points de repère que vous intégrez dans un fichier FLV (lorsque vous créez le fichier FLV), ou que vous créez avec ActionScript .

Points de repère de navigation Vous intégrez des points de repère de navigation dans le flux FLV et le paquet de métadonnées FLV lorsque vous encodez le fichier FLV. Vous utilisez les points de repère de navigation pour permettre aux utilisateurs de rechercher une partie spécifiée d'un fichier.

Points de repère d'événement Vous intégrez des points de repère dans le flux FLV et le paquet de métadonnées FLV lorsque vous encodez le fichier FLV. Vous pouvez rédiger un code pour manipuler les événements qui sont déclenchés à certains points spécifiés pendant la lecture.

Points de repère ActionScript Points de repère externes que vous créez avec le code ActionScript. Vous pouvez rédiger un code pour déclencher ces points de repère en relation avec la lecture vidéo. Ces points de repère sont moins précis que les points de repère intégrés (jusqu'à un dixième de seconde), parce que le lecteur vidéo les suit séparément.

Les points de repère de Navigation créent une image-clé à un emplacement de point de repère spécifié, afin que vous puissiez déplacer la tête de lecture d'un lecteur vidéo à cet emplacement. Vous pouvez définir des points particuliers dans un fichier FLV où vous voulez que les utilisateurs effectuent une recherche. Par exemple, si votre vidéo contient plusieurs chapitres et segments, vous pouvez contrôler la vidéo en intégrant des points de repère de navigation dans le fichier vidéo.

Si vous avez l'intention de créer une application dans laquelle vous voulez pouvoir naviguer jusqu'à un point de repère, vous devez créer et intégrer des points de repère lorsque vous encodez le fichier au lieu d'utiliser les points de repère ActionScript. Vous devez intégrer les points de repère dans le fichier FLV, parce qu'ils sont plus précis. Pour plus d'informations sur le codage de fichiers FLV avec des points de repère, consultez la section « Utilisation des points de repère » du guide *Utilisation de Flash*.

Vous pouvez accéder aux paramètres de points de repère en rédigeant du code ActionScript. Les paramètres de points de repère font partie de l'objet d'événement reçu avec l'événement `cuePoint` (`event.info.parameters`). Pour plus d'informations sur l'encodage de fichiers FLV avec des points de repère, consultez la section « Utilisation des points de repère » du guide *Utilisation de Flash*.

Suivi des points de repère à partir d'un fichier FLV

Vous pouvez suivre les points de repère qui sont intégrés dans un document FLV en utilisant `NetStream.onMetaData`. Vous devez répéter la structure des métadonnées renvoyées pour voir les informations des points de repère.

Le code suivant suit les points de repère dans un fichier FLV :

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
stream_ns.onMetaData = function(metaProp:Object) {
  trace("The metadata:");
  traceMeta(metaProp);
  // traceObject(metaProp, 0);
};
my_video.attachVideo(stream_ns);
stream_ns.play("http://www.helpexamples.com/flash/video/cuepoints.flv");
```

```
function traceMeta(metaProp:Object):Void {
  var p:String;
  for (p in metaProp) {
    switch (p) {
    case "cuePoints" :
      trace("cuePoints: ");
      // passe dans les points de repère
      var cuePointArr:Array = metaProp[p];
      for (var j:Number = 0; j < cuePointArr.length; j++) {
        //passe dans les paramètres courants des points de repère
        trace("\t cuePoints[" + j + "]:");
        var currentCuePoint:Object = metaProp[p][j];
        var metaPropPJParams:Object = currentCuePoint.parameters;
        trace("\t\t name: " + currentCuePoint.name);
        trace("\t\t time: " + currentCuePoint.time);
        trace("\t\t type: " + currentCuePoint.type);
        if (metaPropPJParams != undefined) {
          trace("\t\t parameters:");
          traceObject(metaPropPJParams, 4);
        }
      }
      break;
    default :
      trace(p + ": " + metaProp[p]);
      break;
    }
  }
}
function traceObject(obj:Object, indent:Number):Void {
  var indentString:String = "";
  for (var j:Number = 0; j < indent; j++) {
    indentString += "\t";
  }
  for (var i:String in obj) {
    if (typeof(obj[i]) == "object") {
      trace(indentString + " " + i + ": [Object]");
      traceObject(obj[i], indent + 1);
    } else {
      trace(indentString + " " + i + ": " + obj[i]);
    }
  }
}
```

Le résultat suivant apparaît :

```
The metadata:
canSeekToEnd: true
cuePoints:
  cuePoints[0]:
    name: point1
    time: 0.418
    type: navigation
    parameters:
      lights: beginning
  cuePoints[1]:
    name: point2
    time: 7.748
    type: navigation
    parameters:
      lights: middle
  cuePoints[2]:
    name: point3
    time: 16.02
    type: navigation
    parameters:
      lights: end
audiocodecid: 2
audiodelay: 0.038
audiodatarate: 96
videocodecid: 4
framerate: 15
videodatarate: 400
height: 213
width: 320
duration: 16.334
```

Pour plus d'informations sur l'utilisation des points de repère avec le composant FLVPlayback, consultez la section « Utilisation des points de repère intégrés avec le composant FLVPlayback ».

Utilisation des points de repère intégrés avec le composant FLVPlayback

Vous pouvez afficher les points de repère pour un fichier FLV dans l'inspecteur des propriétés lorsque vous utilisez le composant FLVPlayback. Après avoir défini la propriété `contentPath` pour l'occurrence FLVPlayback, vous pouvez afficher tous les points de repère qui sont intégrés dans le fichier vidéo. Avec l'onglet Paramètre, recherchez la propriété `cuePoints`, puis cliquez sur l'icône en forme de loupe pour afficher une liste des points de repère dans le fichier.

REMARQUE	Pour afficher les points de repère sur l'onglet Paramètre, vous devez saisir le nom de votre fichier FLV dans le champ contentPath au lieu d'utiliser un code pour attribuer le contentPath.

L'exemple suivant indique comment utiliser les informations de points de repère avec le composant FLVPlayback.

Pour utiliser les points de repère avec le composant FLVPlayback :

1. Créez un document Flash appelé **cueFlv.fla**.
2. Ouvrez le panneau Composants (Fenêtre > Composants) et faites glisser une occurrence des composants FLVPlayback et TextArea sur la scène.
3. Choisissez le composant TextArea et tapez **my_ta** dans le champ Nom de l'occurrence dans l'inspecteur des propriétés (Fenêtre > Propriétés > Propriétés).
4. Avec l'occurrence TextArea encore sélectionnée, tapez **200** dans le champ largeur et **100** dans le champ hauteur.
5. Sélectionnez l'occurrence FLVPlayback sur la scène et tapez **my_flvPb** dans le champ Nom de l'occurrence.
6. Sélectionnez l'image 1 du scénario, puis tapez le code suivant dans le panneau Actions :

```
var my_flvPb:mx.video.FLVPlayback;
var my_ta:mx.controls.TextArea;
my_flvPb.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object) {
  my_ta.text += "Elapsed time in seconds: " + my_flvPb.playheadTime +
  "\n";
};
my_flvPb.addEventListener("cuePoint",listenerObject);
```

7. Sélectionnez Contrôle > Tester l'animation pour tester le fichier SWF.

   Le temps écoulé apparaît dans l'occurrence TextArea lorsque la tête de lecteur passe chaque point de repère intégré dans le document.

Pour plus d'informations sur l'utilisation du composant FLVPlayback, consultez la *Référence du langage des composants ActionScript 2.0*.

Création des points de repère avec ActionScript à utiliser avec des composants

Vous pouvez créer des points de repère avec ActionScript, puis les utiliser avec une occurrence d'objet vidéo, ou un des composants du lecteur vidéo (FLVPlayback pour Flash Player 8 et versions ultérieures, ou MediaPlayback pour Flash Player 7). Les exemples suivants montrent combien il est facile d'utiliser du code ActionScript pour créer des points de repère. Ils utilisent ensuite un script pour y accéder.

REMARQUE

Intégrez les points de repère de navigation dans un document si vous avez l'intention d'ajouter la fonctionnalité de navigation à une application. Pour plus d'informations, voir « Utilisation des points de repère », à la page 609. Pour voir un exemple d'utilisation de points de repère intégrés, consultez la section « Utilisation des points de repère intégrés avec le composant FLVPlayback », à la page 613.

Pour créer et utiliser les points de repère avec le composant FLVPlayback :

1. Créez un document Flash appelé **cueFlvPb.fla**.
2. Faites glisser une occurrence du composant FLVPlayback du panneau Composants à la scène (Fenêtre > Composants).

   Le composant est situé dans le dossier FLVPlayback - Player 8.
3. Choisissez le composant et ouvrez l'inspecteur des propriétés (Fenêtre > Propriétés > Propriétés).
4. Tapez **my_flvPb** dans le champ Nom de l'occurrence.
5. Faites glisser une occurrence du composant TextArea du panneau Composants à la scène.
6. Choisissez le composant TextArea et tapez **my_ta** dans le champ Nom de l'occurrence.
7. Avec l'occurrence TextArea encore sélectionnée, tapez **200** dans le champ largeur et **100** dans le champ hauteur.

8. Sélectionnez l'image 1 du scénario, puis tapez le code suivant dans le panneau Actions :

```
var my_flvPb:mx.video.FLVPlayback;
my_flvPb.contentPath = "http://www.helpexamples.com/flash/video/
  clouds.flv";

// Crée un objet cuePoint.
var cuePt:Object = new Object();
cuePt.time = 1;
cuePt.name = "elapsed_time";
cuePt.type = "actionscript";
// Ajoute un point de repère AS.
my_flvPb.addASCuePoint(cuePt);

// Ajoute un autre point de repère AS.
my_flvPb.addASCuePoint(2, "elapsed_time2");

// Affiche les informations de points de repère dans le champ de texte.
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject) {
  my_ta.text += "Elapsed time in seconds: " + my_flvPb.playheadTime +
  "\n";
};
my_flvPb.addEventListener("cuePoint",listenerObject);
```

9. Choisissez Contrôle > Tester l'animation pour tester votre code.

   Les points de repère suivants effectue un tracé dans le panneau Sortie :

```
Elapsed time in seconds: 1.034
Elapsed time in seconds: 2.102
```

Pour plus d'informations sur l'utilisation des points de repère et du composant FLVPlayback, consultez la *Référence du langage des composants ActionScript 2.0*.

L'exemple suivant montre comment ajouter des points de repère à l'exécution, puis les suivre lorsqu'un fichier FLV est lu dans le composant MediaPlayback.

Pour créer et utiliser des points de repère avec le composant MediaPlayback :

1. Créez un document Flash appelé cuePointMP.fla
2. Faites glisser une occurrence du composant MediaPlayback du panneau Composants à la scène (Fenêtre > Composants).

   Le composant est situé dans le dossier Media - Player 6 - 7.
3. Choisissez le composant et ouvrez l'inspecteur des propriétés (Fenêtre > Propriétés > Propriétés).
4. Tapez **my_mp** dans le champ Nom de l'occurrence.
5. Cliquez sur l'onglet Paramètres et cliquez sur Lancer l'inspecteur des composants.
6. Dans l'inspecteur de composants, tapez **http://www.helpexamples.com/flash/video/clouds.flv** dans le champ URL.

7. Ouvrez le panneau Actions (Fenêtre > Actions) et saisissez le code suivant dans le panneau Script :

```
import mx.controls.MediaPlayback;
var my_mp:MediaPlayback;
my_mp.autoPlay = false;
my_mp.addEventListener("cuePoint", doCuePoint);
my_mp.addCuePoint("one", 1);
my_mp.addCuePoint("two", 2);
my_mp.addCuePoint("three", 3);
my_mp.addCuePoint("four", 4);
function doCuePoint(eventObj:Object):Void {
     trace(eventObj.type + " = {cuePointName:" + eventObj.cuePointName +
   " cuePointTime:" + eventObj.cuePointTime + "}");
}
```

8. Choisissez Contrôle > Tester l'animation pour tester votre code.

   Les points de repère suivants effectue un tracé dans le panneau Sortie :

```
cuePoint = {cuePointName:one cuePointTime:1}
cuePoint = {cuePointName:two cuePointTime:2}
cuePoint = {cuePointName:three cuePointTime:3}
cuePoint = {cuePointName:four cuePointTime:4}
```

Pour plus d'informations sur l'utilisation du composant MediaPlayback et du composant FLVPlayback, consultez la *Référence du langage des composants ActionScript 2.0*.

Ajout de la fonctionnalité de recherche avec les points de repère

Vous pouvez intégrer des points de repère de navigation dans un fichier FLV pour ajouter la fonctionnalité de recherche à vos applications. La méthode `seekToNavCuePoint()` du composant FLVPlayback localise le point de repère dans le fichier FLV avec le nom indiqué, au moment spécifié ou après. Vous pouvez spécifier un nom comme chaîne (tel que `"part1"` ou `"theParty"`).

Vous pouvez aussi utiliser la méthode`seekToNextNavCuePoint()`, qui recherche le point de repère de navigation suivant, en fonction du paramètre `playheadTime` actuel. Vous pouvez transmettre à la méthode un paramètre `time`, représentant l'heure de début à partir de laquelle rechercher le point de repère de navigation suivant. La valeur par défaut est le paramètre `playheadTime` actuel.

Sinon, vous pouvez aussi rechercher une durée spécifiée du fichier FLV à l'aide de la méthode `seek()` .

Dans les exemples suivants, vous ajoutez un bouton que vous utilisez pour alterner entre les points de repère ou une durée spécifiée dans un fichier FLV qui est lu dans le composant FLVPlayback, et un bouton pour atteindre un point de repère spécifié.

Pour rechercher à une durée spécifiée :

1. Créez un document Flash appelé **seekduration.fla**.
2. Faites glisser une occurrence du composant FLVPlayback du panneau Composants (Fenêtre > Composants).

   Le composant est dans le dossier FLVPlayback - Player 8.
3. Choisissez le composant et ouvrez l'inspecteur des propriétés (Fenêtre > Propriétés > Propriétés).
4. Tapez **my_flvPb** dans le champ Nom de l'occurrence.
5. Faites glisser une occurrence du composant Button du panneau Composants sur la scène.
6. Choisissez le composant Button et tapez **seek_button** dans le champ Nom de l'occurrence.
7. Sélectionnez l'image 1 du scénario, puis tapez le code suivant dans le panneau Actions :

```
import mx.controls.Button;
import mx.video.FLVPlayback;
var seek_button:Button;
var my_flvPb:FLVPlayback;
my_flvPb.autoPlay = false;
my_flvPb.contentPath = "http://www.helpexamples.com/flash/video/
  sheep.flv";
seek_button.label = "Seek";
seek_button.addEventListener("click", seekFlv);
function seekFlv(eventObj:Object):Void {
  // Recherche à 2 secondes
  my_flvPb.seek(2);
}
```

8. Choisissez Contrôle > Tester l'animation pour tester votre code.

   Lorsque vous cliquez sur le bouton, la tête de lecture vidéo se déplace jusqu'à la durée spécifiée : 2 secondes dans la vidéo.

Pour ajouter la fonctionnalité de recherche avec le composant FLVPlayback :

1. Créez un document Flash appelé **seek1.fla**.
2. Faites glisser une occurrence du composant FLVPlayback du panneau Composants (Fenêtre > Composants).

   Le composant est dans le dossier FLVPlayback - Player 8.
3. Choisissez le composant et ouvrez l'inspecteur des propriétés (Fenêtre > Propriétés > Propriétés).
4. Tapez **my_flvPb** dans le champ Nom de l'occurrence.
5. Faites glisser une occurrence du composant Button du panneau Composants sur la scène.
6. Choisissez le composant Button et tapez **my_button** dans le champ Nom de l'occurrence.

7. Sélectionnez l'image 1 du scénario, puis tapez le code suivant dans le panneau Actions :

```
import mx.video.FLVPlayback;
var my_flvPb:FLVPlayback;
my_flvPb.autoPlay = false;
my_flvPb.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
my_button.label = "Next cue point";

function clickMe(){
  my_flvPb.seekToNextNavCuePoint();
}
my_button.addEventListener("click", clickMe);
```

8. Choisissez Contrôle > Tester l'animation pour tester votre code.

   Le fichier cuepoints.flv contient trois points de repère de navigation : un à côté du début, du milieu et de la fin du fichier vidéo. Lorsque vous cliquez sur le bouton, l'occurrence FLVPlayback recherche le point de repère suivant jusqu'à ce qu'elle atteigne le dernier point de repère dans le fichier vidéo.

Vous pouvez aussi rechercher un point de repère spécifié dans un fichier FLV à l'aide de la méthode `seekToCuePoint()` comme illustré dans l'exemple suivant.

Pour rechercher à un point de repère spécifié :

1. Créez un document Flash appelé **seek2.fla**.
2. Faites glisser une occurrence du composant FLVPlayback du panneau Composants (Fenêtre > Composants).

   Le composant est dans le dossier FLVPlayback - Player 8.
3. Choisissez le composant et ouvrez l'inspecteur des propriétés (Fenêtre > Propriétés > Propriétés).
4. Tapez **my_flvPb** dans le champ Nom de l'occurrence.
5. Avec l'occurrence FLVPlayback encore sélectionnée, cliquez sur l'onglet Paramètres.
6. Tapez **http://www.helpexamples.com/flash/video/cuepoints.flv** dans le champ contentPath.

   Lorsque vous tapez l'URL dans le champ contentPath, les points de repère apparaissent dans l'onglet Paramètres (à côté du paramètre cuePoint). Par conséquent, vous pouvez déterminer le nom du point de repère que vous désirez trouver dans votre code. Si vous cliquez sur l'icône en forme, vous pouvez afficher tous les points de repère du fichier vidéo et les informations à propos de chaque point de repère dans un tableau.
7. Faites glisser une occurrence du composant Button du panneau Composants sur la scène.
8. Choisissez le composant Button et tapez **my_button** dans le champ Nom de l'occurrence.

9. Sélectionnez l'image 1 du scénario, puis tapez le code suivant dans le panneau Actions :

```
import mx.video.FLVPlayback;
var my_flvPb:FLVPlayback;
my_flvPb.autoPlay = false;
my_button.label = "Seek to point2";

function clickMe(){
  my_flvPb.seekToNavCuePoint("point2");
}
my_button.addEventListener("click", clickMe);
```

10. Choisissez Contrôle > Tester l'animation pour tester votre code.

    Le fichier cuepoints.flv contient trois points de repère de navigation : un à côté du début, du milieu et de la fin du fichier vidéo. Lorsque vous cliquez sur le bouton, l'occurrence FLVPlayback recherche au point de repère spécifié (`point2`).

Pour plus d'informations sur les points de repère et le composant FLVPlayback, consultez la *Référence du langage des composants ActionScript 2.0*.

Utilisation des métadonnées

Vous pouvez utiliser la méthode `onMetaData` pour afficher les informations de métadonnées de votre fichier FLV. Les métadonnées comprennent les informations à propos de votre fichier FLV , telles que la durée, la largeur, la hauteur et la cadence d'images. Les informations de métadonnées qui sont ajoutées à votre fichier FLV dépendent du logiciel que vous utilisez pour encoder votre fichier FLV ou du logiciel que vous utilisez pour ajouter les informations de métadonnées.

REMARQUE	Si votre fichier vidéo n'a pas d'information de métadonnées, vous pouvez utiliser des outils pour ajouter des informations de métadonnées à votre fichier.

Pour utiliser `NetStream.onMetaData`, vous devez avoir une vidéo Flash qui contient des métadonnées. Si vous encodez des fichiers FLV avec Flash 8 Video Encoder, votre fichier FLV aura des informations de métadonnées (voir l'exemple suivant pour une liste des métadonnées dans un fichier FLV encodé avec Flash 8 Video Encoder).

REMARQUE	Flash Video Exporter 1.2 et ultérieur (y compris Flash 8 Video Exporter) ajoutent des métadonnées à vos fichiers FLV. Sorenson Squeeze 4.1 et ultérieur ajoutent aussi des métadonnées à vos fichiers FLV.

L'exemple suivant utilise `NetStream.onMetaData` pour suivre les informations de métadonnées d'un fichier FLV encodé avec Flash 8 Video Encoder.

Pour utiliser NetStream.onMetaData pour afficher les informations de métadonnées :

1. Créez un fichier FLA appelé **flvMetadata.fla**.
2. Dans le panneau Bibliothèque (Fenêtre > Bibliothèque), choisissez Nouvelle vidéo dans le menu contextuel Bibliothèque.
3. Dans la boîte de dialogue Propriétés vidéo, nommez le symbole vidéo et sélectionnez Vidéo (contrôlé par ActionScript).
4. Cliquez sur OK pour créer un objet vidéo.
5. Faites glisser l'objet vidéo du panneau Bibliothèque sur la scène pour créer une occurrence d'objet vidéo.
6. Lorsque l'objet vidéo est sélectionné sur la scène, dans l'inspecteur des propriétés (Fenêtre > Propriétés > Propriétés), saisissez **my_video** dans le champ de texte Nom de l'occurrence.
7. Avec l'occurrence vidéo encore sélectionnée, tapez **320** dans le champ largeur et **213** dans le champ hauteur.
8. Sélectionnez l'image 1 dans le scénario et ouvrez le panneau Actions (Fenêtre > Actions).
9. Tapez le code suivant dans le panneau Actions :

```
// Créer un objet NetConnection.
var netConn:NetConnection = new NetConnection();
// Crée une connexion locale en flux continu
netConn.connect(null);
// Crée un objet NetStream et définit une fonction onStatus()
var nStream:NetStream = new NetStream(netConn);
// Associe la source vidéo NetStream à l'objet Video
my_video.attachVideo(nStream);
// Définit la durée du tampon
nStream.setBufferTime(30);
// Lit le fichier FLV.
nStream.play("http://www.helpexamples.com/flash/video/
  lights_short.flv");
// Suit les métadonnées.
nStream.onMetaData = function(myMeta) {
     for (var i in myMeta) {
          trace(i + ":\t" + myMeta[i])
     }
};
```

10. Choisissez Contrôle > Tester l'animation pour tester votre code.

Les informations suivantes apparaissent dans le panneau de sortie :

```
canSeekToEnd:true
audiocodecid:2
audiodelay:0.038
audiodatarate:96
videocodecid:4
framerate:15
videodatarate:400
height:213
width:320
duration:8.04
```

REMARQUE Si la vidéo ne contient pas de son, les informations de métadonnées associées à l'audio (telles que `audiodatarate`) renvoient `undefined`, car aucune information audio n'est ajoutée aux métadonnées pendant l'encodage.

Vous pouvez également utiliser le format suivant pour afficher la plupart des informations de métadonnées. Par exemple, le code suivant montre la durée d'un fichier FLV :

```
nStream.onMetaData = function(myMeta) {
  trace("FLV duration: " + myMeta.duration + " sec.");
};
```

Ce format ne peut pas suivre l'information de métadonnées `cuePoint`. Pour plus d'informations sur le suivi des points de repère, consultez la section « Suivi des points de repère à partir d'un fichier FLV », à la page 610.

Configuration de votre serveur pour les fichiers FLV

Pour utiliser des fichiers FLV, il peut être nécessaire de configurer votre serveur pour le format de fichier FLV. le protocole MIME (Multipurpose Internet Mail Extensions) est une spécification de données normalisée qui vous permet d'envoyer des fichiers non-ASCII files sur des connexions Internet. Les navigateurs Web et les clients de courrier électronique sont configurés pour interpréter de nombreux *types MIME* afin qu'ils puissent envoyer et recevoir de la vidéo, de l'audio, des graphiques et du texte formaté. Pour charger des fichiers FLV à partir d'un serveur Web, il peut être nécessaire d'enregistrer l'extension de fichier et le type MIME auprès de votre serveur Web. Pour plus d'informations, consultez la documentation du serveur. Le type MIME des fichiers FLV est `video/x-flv`. Les informations complètes du type de fichier FLV se présentent comme suit :

Type MIME : video/x-flv

extensions de fichier : .flv

Paramètres requis : aucun

Paramètres optionnels : aucun

Considérations sur l'encodage : Les fichiers FLV sont binaires et une partie des applications peuvent nécessiter la définition du sous-type application/flux d'octets.

Problèmes de sécurité : aucun

Spécification publiée : www.adobe.com/go/flashfileformat_fr.

Microsoft a changé la façon dont le média en flux continu est géré par le serveur Web 6.0 IIS (Microsoft Internet Information Services) par rapport à ses versions antérieures. Les versions antérieures d'IIS ne nécessitent aucune modification pour diffuser en continu la vidéo Flash. Dans IIS 6.0, le serveur Web fourni par défaut avec Windows 2003, le serveur nécessite un type MIME pour reconnaître que les fichiers FLV sont des médias à flux continu.

Lorsque des fichiers SWF qui diffusent des fichiers FLV externes sont placés sur un serveur Microsoft Windows 2003 et sont affichés dans un navigateur, le fichier SWF est lu correctement, mais la vidéo FLV n'est pas diffusée en continu. Ce problème affecte tous les fichiers FLV placés sur le serveur Windows 2003, y compris les fichiers créés avec des versions antérieures de l'outil de programmation de Flash, le Kit Macromedia Flash Video pour Dreamweaver MX 2004. Ces fichiers fonctionnent correctement si vous les testez sur d'autres systèmes d'exploitation.

Pour plus d'informations sur la manière de configurer Microsoft Windows 2003 et Microsoft IIS Server 6.0 pour diffuser en continu la vidéo FLV, consultez la page http://www.adobe.com/cfusion/knowledgebase/index.cfm?id=tn_19439.

Ciblage des fichiers FLV locaux sur Macintosh

Si vous tentez de lire une vidéo FLV locale à partir d'un lecteur non système sur un ordinateur Macintosh avec un chemin qui utilise une barre oblique (/), elle ne sera pas lue. *Les lecteurs non système* sont par exemple les CD-ROM, les disques durs partitionnés, les supports de stockage amovibles, les périphériques de stockage connectés (la liste est incomplète).

REMARQUE

La raison de cet échec est une limitation du système d'exploitation, et non pas une limitation de Flash ou de Flash Player.

Pour qu'un fichier FLV soit lu à partir d'un lecteur non système sur un Macintosh, faites-y référence à l'aide d'un chemin absolu utilisant une notation à deux points (:) au lieu d'une notation à barres obliques. La liste suivante montre la différence entre les deux sortes de notation :

Notation à barres obliques myDrive/myFolder/myFLV.flv

Notation à deux points (Macintosh) myDrive:myFolder:myFLV.flv

Vous pouvez aussi créer un fichier de projection pour un CD-ROM que vous voulez utiliser pour la lecture Macintosh. Pour obtenir les dernières informations sur les CD-ROM Macintosh et les fichiers FLV, consultez la page http://www.adobe.com/cfusion/knowledgebase/index.cfm?id=3121b301.

Création d'animation de progression pour les fichiers média

ActionScript offre plusieurs manières de précharger un fichier de média externe ou de suivre la progression de son téléchargement. Vous pouvez créer des barres ou des animations de progression pour afficher la progression du téléchargement ou la quantité de contenu qui a été chargé.

Pour précharger des fichiers SWF et JPEG, utilisez la classe MovieClipLoader, qui offre un mécanisme d'écouteur d'événements permettant de vérifier la progression du téléchargement. Pour plus d'informations, consultez la section « Préchargement de fichiers SWF et JPEG », page 427.

Pour suivre la progression du téléchargement des fichiers MP3, utilisez les méthodes `Sound.getBytesLoaded()` et `Sound.getBytesTotal()` ; pour suivre la progression du téléchargement des fichiers FLV, utilisez les propriétés `NetStream.bytesLoaded` et `NetStream.bytesTotal`. Pour plus d'informations, consultez la section « Préchargement de fichiers MP3 », page 420.

Pour plus d'informations sur la création de barres de progression pour charger des fichiers de médias, consultez les rubriques suivantes :

- « Création d'une animation de progression pour le chargement de fichiers SWF et de fichiers d'images », à la page 624
- « Création d'une barre de progression pour le téléchargement de fichiers MP3 avec ActionScript », à la page 626
- « Création d'une barre de progression pour le téléchargement de fichiers FLV avec ActionScript », à la page 628

Pour un exemple de fichier source, tweenProgress.fla, qui utilise une animation programmée pour créer une animation de barre de progression, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Tween Progressbar afin d'accéder à cet exemple.

Création d'une animation de progression pour le chargement de fichiers SWF et de fichiers d'images

Lorsque vous chargez des fichiers SWF ou des fichiers d'images volumineux dans une application, vous pouvez avoir besoin d'une animation qui affiche la progression du chargement. Vous pouvez créer une barre de progression qui affiche l'état d'avancement de l'opération pendant que l'animation se charge. Vous pouvez aussi créer une animation qui change tandis que le fichier se charge. Pour plus d'informations sur le chargement des fichiers SWF et des fichiers d'images, consultez la section « Chargement de fichiers SWF et de fichiers d'images externes », à la page 591.

L'exemple suivant montre comment utiliser la classe MovieClipLoader et l'API de dessin pour afficher la progression du téléchargement d'un fichier image.

Pour créer une barre de progression pour le téléchargement de fichiers d'images ou de fichiers SWF :

1. Créez un document Flash appelé **loadImage.fla**.
2. Choisissez Modifier > Document, et tapez **700** dans le champ largeur et **500** dans le champ hauteur pour changer les dimensions du document.

3. Sélectionnez l'image 1 du scénario, puis tapez le code suivant dans le panneau Actions :

```
// Crée des clips pour recevoir votre contenu
this.createEmptyMovieClip("progressBar_mc", 0);
progressBar_mc.createEmptyMovieClip("bar_mc", 1);
progressBar_mc.createEmptyMovieClip("stroke_mc", 2);
//Utilise des méthodes de dessin pour créer une barre de progression.
with (progressBar_mc.stroke_mc) {
   lineStyle(0, 0x000000);
   moveTo(0, 0);
   lineTo(100, 0);
   lineTo(100, 10);
   lineTo(0, 10);
   lineTo(0, 0);
}
with (progressBar_mc.bar_mc) {
   beginFill(0xFF0000, 100);
   moveTo(0, 0);
   lineTo(100, 0);
   lineTo(100, 10);
   lineTo(0, 10);
   lineTo(0, 0);
   endFill();
   _xscale = 0;
}
progressBar_mc._x = 2;
progressBar_mc._y = 2;
// Progression du téléchargement
var mclListener:Object = new Object();
mclListener.onLoadStart = function(target_mc:MovieClip) {
   progressBar_mc.bar_mc._xscale = 0;
};
mclListener.onLoadProgress = function(target_mc:MovieClip,
  bytesLoaded:Number, bytesTotal:Number) {
   progressBar_mc.bar_mc._xscale = Math.round(bytesLoaded/
  bytesTotal*100);
};
mclListener.onLoadComplete = function(target_mc:MovieClip) {
   progressBar_mc.removeMovieClip();
};
mclListener.onLoadInit = function(target_mc:MovieClip) {
   target_mc._height = 500;
   target_mc._width = 700;
};
// Crée un clip parent pour le recevoir l'image.
this.createEmptyMovieClip("image_mc", 100);
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
/* Charge l'image dans le clip.
Vous pouvez modifier l'URL suivante vers un fichier SWF ou un autre
  fichier d'image. */
image_mcl.loadClip("http://www.helpexamples.com/flash/images/gallery1/
  images/pic3.jpg", image_mc);
```

4. Choisissez Contrôle > Tester l'animation pour visualiser le chargement de l'image et la barre de progression.

REMARQUE	Si vous testez ce code une seconde fois, l'image sera mise en cache et la barre de progression se terminera immédiatement. Pour effectuer des tests multiples, utilisez différentes images et chargez-les à partir d'une source externe. Une source locale peut causer des problèmes avec le test de votre application parce que le contenu se charge trop rapidement.

Pour un exemple de fichier source, tweenProgress.fla, qui utilise une animation programmée pour créer une animation de barre de progression, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Tween Progressbar afin d'accéder à cet exemple.

Pour des exemples d'applications de galerie de photos, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Galleries afin d'accéder à ces exemples.

- gallery_tree.fla
- gallery_tween.fla

Ces exemples vous montrent comment utiliser le code ActionScript pour contrôler dynamiquement des clips tout en chargeant des fichiers image dans un fichier SWF.

Création d'une barre de progression pour le téléchargement de fichiers MP3 avec ActionScript

L'exemple suivant charge plusieurs morceaux dans un fichier SWF. Une barre de progression, créée avec l'API de dessin, indique la progression du téléchargement. Lorsque la musique commence et termine son chargement, des informations s'affichent dans le panneau de sortie. Pour plus d'informations sur le chargement de fichiers MP3, consultez la section « Chargement d'un fichier MP3 », à la page 597.

Pour créer une barre de progression pour le téléchargement de fichiers MP3 :

1. Créez un document Flash appelé **loadSound.fla**.
2. Sélectionnez l'image 1 du scénario, puis tapez le code suivant dans le panneau Actions :

```
var pb_height:Number = 10;
var pb_width:Number = 100;
var pb:MovieClip = this.createEmptyMovieClip("progressBar_mc",
  this.getNextHighestDepth());
pb.createEmptyMovieClip("bar_mc", pb.getNextHighestDepth());
pb.createEmptyMovieClip("vBar_mc", pb.getNextHighestDepth());
pb.createEmptyMovieClip("stroke_mc", pb.getNextHighestDepth());
```

```
pb.createTextField("pos_txt", pb.getNextHighestDepth(), 0, pb_height,
  pb_width, 22);

pb._x = 100;
pb._y = 100;

with (pb.bar_mc) {
   beginFill(0x00FF00);
   moveTo(0, 0);
   lineTo(pb_width, 0);
   lineTo(pb_width, pb_height);
   lineTo(0, pb_height);
   lineTo(0, 0);
   endFill();
   _xscale = 0;
}
with (pb.vBar_mc) {
   lineStyle(1, 0x000000);
   moveTo(0, 0);
   lineTo(0, pb_height);
}
with (pb.stroke_mc) {
   lineStyle(3, 0x000000);
   moveTo(0, 0);
   lineTo(pb_width, 0);
   lineTo(pb_width, pb_height);
   lineTo(0, pb_height);
   lineTo(0, 0);
}

var my_interval:Number;
var my_sound:Sound = new Sound();
my_sound.onLoad = function(success:Boolean) {
   if (success) {
      trace("sound loaded");
   }
};
my_sound.onSoundComplete = function() {
   clearInterval(my_interval);
   trace("Cleared interval");
}
my_sound.loadSound("http://www.helpexamples.com/flash/sound/song2.mp3",
  true);
my_interval = setInterval(updateProgressBar, 100, my_sound);

function updateProgressBar(the_sound:Sound):Void {
   var pos:Number = Math.round(the_sound.position / the_sound.duration *
  100);
   pb.bar_mc._xscale = pos;
   pb.vBar_mc._x = pb.bar_mc._width;
   pb.pos_txt.text = pos + "%";
}
```

3. Choisissez Contrôle > Tester l'animation pour charger le fichier MP3 et visualiser la barre de progression.

REMARQUE	Si vous testez ce code une seconde fois, l'image sera mise en cache et la barre de progression se terminera immédiatement. Pour effectuer des tests multiples, utilisez différentes images et chargez-les à partir d'une source externe. Une source locale peut causer des problèmes avec le test de votre application parce que le contenu se charge trop rapidement.

Pour plus d'informations sur l'utilisation de son, consultez l'entrée de la classe Sound, `Sound`, dans le *Guide de référence du langage ActionScript 2.0*.

Pour un exemple de fichier source, tweenProgress.fla, qui utilise une animation programmée pour créer une animation de barre de progression, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Tween Progressbar afin d'accéder à cet exemple.

Pour un exemple de fichier source, jukebox.fla, qui charge des fichiers MP3, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Jukebox afin d'accéder à cet exemple. Cet exemple montre comment créer un juke-box en utilisant les types de données, les principes de codage général et plusieurs composants.

Création d'une barre de progression pour le téléchargement de fichiers FLV avec ActionScript

Vous pouvez créer une barre de progression pour afficher la progression du téléchargement d'un fichier FLV. Pour plus d'informations sur le chargement de fichiers FLV dans un fichier SWF, consultez la section « Préchargement de fichiers FLV », à la page 608. Pour d'autres d'informations sur les fichiers FLV et Flash, consultez la section « Utilisation du format vidéo FLV », à la page 601.

L'exemple suivant utilise l'API de dessin pour créer une barre de progression. L'exemple suivant utilise aussi les propriétés `bytesLoaded` et `bytesTotal`, qui affichent la progression du téléchargement de video1.flv dans l'occurrence d'objet vidéo appelée `my_video`. Un champ `loaded_txt` est créé de façon dynamique pour afficher des informations sur la progression du téléchargement.

Pour créer une barre de progression qui affiche la progression du téléchargement :

1. Créez un fichier FLA appelé **flvProgress.fla**.
2. Dans le panneau Bibliothèque (Fenêtre > Bibliothèque), choisissez Nouvelle vidéo dans le menu contextuel Bibliothèque.
3. Dans la boîte de dialogue Propriétés vidéo, nommez le symbole vidéo et sélectionnez Vidéo (contrôlé par ActionScript).
4. Cliquez sur OK pour créer un objet vidéo.
5. Faites glisser l'objet vidéo du panneau Bibliothèque sur la scène pour créer une occurrence d'objet vidéo.
6. Lorsque l'objet vidéo est sélectionné sur la scène, dans l'inspecteur des propriétés (Fenêtre> Propriétés> Propriétés), saisissez **my_video** dans le champ de texte Nom de l'occurrence.
7. Avec l'occurrence vidéo encore sélectionnée, tapez **320** dans le champ largeur et **213** dans le champ hauteur.
8. Sélectionnez l'image 1 du scénario, puis tapez le code suivant dans le panneau Actions :

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("http://www.helpexamples.com/flash/video/
  typing_short.flv");

this.createTextField("loaded_txt", this.getNextHighestDepth(), 10, 10,
  160, 22);
this.createEmptyMovieClip("progressBar_mc", this.getNextHighestDepth());
progressBar_mc.createEmptyMovieClip("bar_mc",
  progressBar_mc.getNextHighestDepth());
with (progressBar_mc.bar_mc) {
   beginFill(0xFF0000);
   moveTo(0, 0);
   lineTo(100, 0);
   lineTo(100, 10);
   lineTo(0, 10);
   lineTo(0, 0);
   endFill();
   _xscale = 0;
}
```

```
progressBar_mc.createEmptyMovieClip("stroke_mc",
  progressBar_mc.getNextHighestDepth());
with (progressBar_mc.stroke_mc) {
   lineStyle(0, 0x000000);
   moveTo(0, 0);
   lineTo(100, 0);
   lineTo(100, 10);
   lineTo(0, 10);
   lineTo(0, 0);
}

var loaded_interval:Number = setInterval(checkBytesLoaded, 500,
  stream_ns);
function checkBytesLoaded(my_ns:NetStream) {
   var pctLoaded:Number = Math.round(my_ns.bytesLoaded /
  my_ns.bytesTotal * 100);
   loaded_txt.text = Math.round(my_ns.bytesLoaded / 1000) + " of " +
  Math.round(my_ns.bytesTotal / 1000) + " KB loaded (" + pctLoaded +
  "%)";
   progressBar_mc.bar_mc._xscale = pctLoaded;
   if (pctLoaded>=100) {
      clearInterval(loaded_interval);
   }
}
```

9. Choisissez Contrôle > Tester l'animation pour tester votre code.

La vidéo se charge et une barre d'animation et des valeurs de texte changeantes communique la progression de chargement. Si ces éléments chevauchent votre vidéo, déplacez l'objet vidéo sur la scène. Vous pouvez personnaliser la couleur de la barre de progression en modifiant `beginFill` et `lineStyle` dans le code précédent.

REMARQUE Si votre barre de progression se charge instantanément, c'est que la vidéo a déjà été mise en cache sur votre disque dur (à partir du test de cet exemple ou lors du chargement d'une procédure différente). Dans pareil cas, téléchargez un fichier FLV sur votre serveur et chargez-le à la place.

Pour un exemple de fichier source, tweenProgress.fla, qui utilise une animation programmée pour créer une animation de barre de progression, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Tween Progressbar afin d'accéder à cet exemple.

CHAPITRE 15

Utilisation de données externes

15

Dans Flash CS3 Professional, vous pouvez utiliser ActionScript pour charger des données provenant de sources externes dans un fichier SWF. Vous pouvez aussi envoyer des données, qui peuvent être fournies par l'utilisateur sur le serveur, à partir d'un fichier SWF vers un serveur d'application (comme ColdFusion ou JRun) ou par un autre type de script côté serveur comme PHP ou Perl. Flash Player peut envoyer et charger des données sur HTTP, HTTPS ou à partir d'un fichier texte local. Vous pouvez également créer des connexions socket TCP/IP durables pour des applications à faible temps d'attente (ex. : applications de discussions en ligne ou de cotations boursières). Flash Player 8 et versions ultérieures prennent maintenant en charge le téléchargement de fichiers d'un ordinateur d'utilisateur vers un serveur et vice-versa.

Il est possible de formater les données que vous chargez ou que vous envoyez à partir d'un fichier SWF sous forme d'un fichier XML (Extensible Markup Language) ou de paires nom-valeur.

Flash Player peut également échanger des données avec son environnement hôte (un navigateur Web, par exemple) ou avec une autre occurrence de Flash Player sur le même ordinateur ou la même page Web.

Par défaut, un fichier SWF ne peut accéder qu'aux données qui résident exactement dans le même domaine (par exemple, www.adobe.com). (Pour plus d'informations, voir « Restriction des API de réseau », à la page 695).

Pour plus d'informations sur l'utilisation de données externes, consultez les rubriques suivantes :

Envoi et chargement des variables

Un fichier SWF est une fenêtre destinée à récupérer et à afficher des informations, un peu comme une page HTML. Cependant, les fichiers SWF peuvent rester chargés dans le navigateur et être mis à jour en permanence à l'aide de nouvelles informations sans qu'il soit nécessaire d'actualiser toute la page. Grâce aux fonctions et méthodes d'ActionScript, vous pouvez envoyer et recevoir des informations à partir de scripts côté serveur et recevoir des informations en provenance de fichiers texte et de fichiers XML .

En outre, les scripts côté serveur peuvent demander des informations précises à une base de données et les transmettre à un fichier SWF. Les scripts côté serveur peuvent être rédigés en plusieurs langages, les plus communs étant CFML, Perl, ASP (Microsoft Active Server Pages) et PHP. Le stockage et l'extraction des informations par le biais d'une base de données vous permet de créer un contenu dynamique et personnalisé pour votre fichier SWF. Par exemple, vous pouvez créer un panneau d'affichage, des profils utilisateur personnels ou un caddie permettant d'effectuer le suivi des achats d'un utilisateur.

Plusieurs fonctions et méthodes d'ActionScript vous permettent d'échanger des informations avec un fichier SWF. Chaque fonction ou chaque méthode utilise un protocole pour transférer les informations, dont le format doit être spécifique.

- Les fonctions et les méthodes MovieClip utilisant le protocole HTTP ou HTTPS pour envoyer des informations au format de code URL sont `getURL()`, `loadVariables()`, `loadVariablesNum()`, `loadMovie()` et `loadMovieNum()`.
- Les méthodes LoadVars utilisant le protocole HTTP ou HTTPS pour échanger des informations au format de code URL sont `load()`, `send()` et `sendAndLoad()`.
- Les méthodes utilisant le protocole HTTP ou HTTPS pour échanger des informations sous forme de fichiers XML sont `XML.send()`, `XML.load()` et `XML.sendAndLoad()`.
- Les méthodes qui créent et utilisent une connexion socket TCP/IP pour échanger des informations sous forme de fichiers XML sont `XMLSocket.connect()` et `XMLSocket.send()`.

Pour plus d'informations, consultez les sections suivantes :

- « Vérification des données chargées », à la page 633
- « Création d'une barre de progression affichant l'avancée du chargement des données », à la page 634

Vérification des données chargées

Toute fonction ou méthode qui charge des données vers un fichier SWF (sauf `XMLSocket.send()`) est *asynchrone*. les résultats de l'action sont renvoyés à un moment indéterminé.

Avant de pouvoir utiliser les données chargées dans un fichier SWF, vous devez d'abord vérifier si elles ont bien été chargées. Par exemple, vous ne pouvez pas charger de variables et manipuler leurs valeurs dans le même script, car ces données n'existent pas dans le fichier tant que ce dernier n'a pas été chargé. Dans le script suivant, vous ne pouvez pas utiliser la variable `lastSiteVisited` tant que vous n'êtes pas certain que la variable a été chargée depuis le fichier myData.txt, vous aurez un texte semblable à l'exemple suivant :

```
lastSiteVisited=www.adobe.com
```

Mais si vous avez utilisé le code suivant, vous n'avez pas pu suivre les données en cours de chargement :

```
loadVariables("myData.txt", 0);
trace(lastSiteVisited); // non défini
```

Chaque fonction ou méthode possède une technique spécifique que vous pouvez utiliser pour vérifier les données qui ont été chargées. Si vous utilisez la fonction loadVariables ou la fonction loadMovie, vous pouvez charger des informations dans un clip cible et utiliser le gestionnaire `onData` pour exécuter un script. Si vous utilisez la fonction loadVariables pour charger les données, le gestionnaire `onData` s'exécute une fois la dernière variable chargée. Si vous utilisez la fonction loadMovie pour charger les données, le gestionnaire `onData` s'exécute chaque fois qu'une partie du fichier SWF est transmise à Flash Player.

Par exemple, le code ActionScript suivant charge les variables du fichier myData.txt dans le clip `loadTarget_mc`. un gestionnaire `onData()` affecté à l'occurrence `loadTarget_mc` utilise la variable `lastSiteVisited`, chargée depuis le fichier myData.txt. Les actions trace suivantes apparaissent uniquement après que toutes les variables, y compris `lastSiteVisited`, sont chargées :

```
this.createEmptyMovieClip("loadTarget_mc", this.getNextHighestDepth());
this.loadTarget_mc.onData = function() {
  trace("Data Loaded");
  trace(this.lastSiteVisited);
};
loadVariables("myData.txt", this.loadTarget_mc);
```

Si vous utilisez les méthodes `XML.load()`, `XML.sendAndLoad()` et `XMLSocket.connect()`, vous devez définir un gestionnaire qui traite les données dès leur arrivée. Ce gestionnaire est une propriété d'un objet XML ou XMLSocket auquel vous affectez une fonction que vous avez définie. Les gestionnaires sont appelés automatiquement lorsque les informations sont reçues. Pour l'objet XML, utilisez `XML.onLoad()` ou `XML.onData()`. Pour l'objet XMLSocket, utilisez `XMLSocket.onConnect()`.

Pour plus d'informations, voir « Utilisation de la classe XML », à la page 653 et « Utilisation de la classe XMLSocket », à la page 659. Pour plus d'informations sur l'utilisation de LoadVars pour échanger des données qui peuvent être traitées après leur réception, consultez la section « Utilisation de la classe LoadVars », à la page 638.

Création d'une barre de progression affichant l'avancée du chargement des données

L'exercice suivant crée dynamiquement un préchargement simple en utilisant l'interface de programmation d'application (API) de dessin et affiche la progression du chargement pour un document XML.

CONSEIL	Si le fichier XML à distance se charge trop rapidement pour voir l'effet de préchargement, essayez de télécharger un plus grand fichier XML vers Internet et télécharger ce fichier.

Créer une barre de progression avec l'API de dessin

1. Créez un document Flash, puis enregistrez-le sous le nom de **drawapi.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
var barWidth:Number = 200;
var barHeight:Number = 6;

this.createEmptyMovieClip("pBar_mc", 9999);
var bar:MovieClip = pBar_mc.createEmptyMovieClip("bar_mc", 10);
bar.beginFill(0xFF0000, 100);
bar.moveTo(0, 0);
bar.lineTo(barWidth, 0);
bar.lineTo(barWidth, barHeight);
bar.lineTo(0, barHeight);
bar.lineTo(0, 0);
bar.endFill();
bar._xscale = 0;
```

```
var stroke:MovieClip = pBar_mc.createEmptyMovieClip("stroke_mc", 20);
stroke.lineStyle(0, 0x000000);
stroke.moveTo(0, 0);
stroke.lineTo(barWidth, 0);
stroke.lineTo(barWidth, barHeight);
stroke.lineTo(0, barHeight);
stroke.lineTo(0, 0);

pBar_mc.createTextField("label_txt", 30, 0, barHeight, 100, 21);
pBar_mc.label_txt.autoSize = "left";
pBar_mc.label_txt.selectable = false;

pBar_mc._x = (Stage.width - pBar_mc._width) / 2;
pBar_mc._y = (Stage.height - pBar_mc._height) / 2;

var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean) {
  pBar_mc.onEnterFrame = undefined;
  if (success) {
    trace("XML loaded successfully");
  } else {
    trace("Unable to load XML");
  }
};
my_xml.load("http://www.helpexamples.com/flash/xml/ds.xml");

pBar_mc.onEnterFrame = function() {
  var pctLoaded:Number = Math.floor(my_xml.getBytesLoaded() /
  my_xml.getBytesTotal() * 100);
  if (!isNaN(pctLoaded)) {
    pBar_mc.bar_mc._xscale = pctLoaded;
    pBar_mc.label_txt.text = pctLoaded + "% loaded";
    if (pctLoaded >= 100) {
      pBar_mc.onEnterFrame = undefined;
    }
  }
};
```

Le code précédent est divisé en sept sections. La première section définit la largeur et la hauteur de la barre de progression lorsqu'elle est dessinée sur la scène. La barre de progression sera centrée sur la scène dans une section ultérieure. La section de code suivante crée deux clips vidéo, `pBar_mc` and `bar_mc`. le clip `bar_mc` est imbriqué dans `pBar_mc`, et dessine un rectangle rouge sur la scène. L'occurrence `bar_mc` modifie sa propriété `_xscale` tandis que le fichier XML externe se charge à partir du site Web à distance.

Ensuite, un second clip vidéo est imbriqué dans le clip `pBar_mc`, `stroke_mc`. Le clip `stroke_mc` dessine un contour sur la scène qui correspond aux dimensions spécifiées parles variables `barHeight` et `barWidth` définies dans la première section. La quatrième section de code crée dans le clip `pBar_mc` un champ de texte qui est utilisé pour afficher quel pourcentage du fichier XML a déjà été chargé, semblable à l'étiquette sur le composant ProgressBar. Ensuite, le clip `pBar_mc` (qui comprend les occurrences imbriquées `bar_mc`, `stroke_mc`, et `label_txt`) est centré sur la scène.

La sixième section de code définit une nouvelle occurrence d'objet XML qui sert à charger un fichier XML externe. Un gestionnaire d'événement onLoad est défini et suit un message vers le panneau de sortie. Le gestionnaire d'événement onLoad supprime aussi le gestionnaire d'événement`onEnterFrame` (qui est défini dans la section suivante pour le clip `pBar_mc`. La section finale de code définit un gestionnaire d'événement `onEnterFrame` pour le clip `pBar_mc`. Ce gestionnaire d'événement surveille combien du fichier XML externe a été chargé et modifie la propriété `_xscale` pour le clip `bar_mc`. D'abord le gestionnaire d'événement `onEnterFrame` calcule le pourcentage du fichier qui a terminé son téléchargement. Aussi longtemps que le pourcentage du fichier chargé est un nombre valide, la propriété `_xscale` pour `bar_mc` est définie, et le champ de texte dans `pBar_mc` affiche quel pourcentage du fichier a été chargé. Si le fichier a terminé son téléchargement (le pourcentage chargé atteint 100%) le gestionnaire d'événement `onEnterFrame` est supprimé et donc la progression du téléchargement n'est plus surveillée.

3. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

   Tandis que le fichier XML externe se charge, le clip `bar_mc` imbriqué se redimensionne pour afficher la progression du téléchargement du XML. Une fois que le fichier XML a terminé son téléchargement, le gestionnaire d'événement `onEnterFrame` est supprimé afin qu'il ne continue pas de calculer la progression du téléchargement. Selon la vitesse à laquelle le téléchargement se termine, vous devez pouvoir voir la barre grandir lentement jusqu'à ce que `bar_mc` soit de la même largeur que le clip `stroke_mc`. Si le téléchargement est trop rapide, la barre de progression peut aller de 0% à 100% trop vite, rendant l'effet plus difficile à voir. Dans ce cas il peut être nécessaire de tenter le téléchargement d'un plus grand fichier XML.

Utilisation du protocole HTTP pour les connexions aux scripts côté serveur

Les fonctions loadVariables, loadVariablesNum, getURL, loadMovie, loadMovieNum et les méthodes loadVariables (méthode MovieClip.loadVariables), loadMovie (méthode MovieClip.loadMovie) et getURL (méthode MovieClip.getURL) peuvent communiquer avec des scripts côté serveur à l'aide des protocoles HTTP ou HTTPS.Ces fonctions et méthodes envoient toutes les variables provenant du scénario auquel la fonction est attachée. Lorsqu'elles sont utilisées comme méthodes de l'objet MovieClip, les fonctions `loadVariables()`, `getURL()` et `loadMovie()` envoient toutes les variables du clip indiqué ; chaque fonction (ou méthode) traite sa réponse de la manière suivante :

- La fonction `getURL()` renvoie les informations dans une fenêtre de navigateur et non dans Flash Player.
- La méthode `loadVariables()` charge les variables dans un scénario ou un niveau spécifié de Flash Player.
- La méthode `loadMovie()` charge un fichier SWF dans un clip ou niveau spécifié dans Flash Player.

Lorsque vous utilisez `loadVariables()`, `getURL()` ou `loadMovie()`, vous pouvez spécifier plusieurs paramètres :

- *`URL`* est le fichier dans lequel se trouvent les variables distantes.
- *`Emplacement`* est le niveau ou la cible dans le fichier SWF qui reçoit les variables. (La fonction `getURL` ne prend pas ce paramètre.)

  Pour plus d'informations sur les niveaux et les cibles, consultez le guide *Utilisation de Flash*.
- *`Variables`* définit la méthode HTTP, `GET` (ajoute les variables à la fin de l'URL) ou `POST` (place les variables dans un en-tête HTTP distinct). Les variables sont envoyées par l'intermédiaire de cette méthode. Lorsque ce paramètre est omis, Flash Player utilise par défaut la méthode `GET`, mais aucune variable n'est envoyée.

Par exemple, pour suivre les meilleurs scores d'un jeu, vous pouvez stocker les scores sur un serveur et utiliser une fonction `loadVariables()` pour les charger dans le fichier SWF chaque fois que quelqu'un joue à ce jeu. L'appel de fonction pourrait avoir l'aspect suivant :

```
this.createEmptyMovieClip("highscore_mc", 10);
loadVariables("http://www.helpexamples.com/flash/highscore.php",
  highscore_mc, "GET");
```

Cet exemple charge les variables du script ColdFusion high_score.cfm dans l'occurrence de clip `scoreClip` en utilisant la méthode HTTP `GET`.

Les variables chargées à l'aide de la fonction `loadVariables()` doivent être au format MIME standard *application/x-www-form-urlencoded* (un format standard utilisé par les scripts CFM et CGI). Le fichier que vous spécifiez dans le paramètre d'*URL* de `loadVariables()` doit écrire les paires variable et valeur dans ce format afin que Flash puisse les lire. Ce fichier peut spécifier n'importe quel nombre de variables ; les paires variable et valeur doivent être séparées avec une esperluette (`&`), et les mots au sein d'une valeur doivent être séparés par un signe plus (+). Par exemple, cette séquence définit plusieurs variables :

```
highScore1=54000&playerName1=RGoulet&highScore2=53455&playerName2=
  WNewton&highScore3=42885&playerName3=TJones
```

REMARQUE

Il peut être nécessaire de coder certains caractères pour l'URL, tels que les signes (+) ou éperluette (&). Pour plus d'informations, visitez le site www.adobe.com/go/tn_14143_fr.

Pour plus d'informations, consultez la section suivante : « Utilisation de la classe LoadVars », à la page 638. Consultez également les fonctions loadVariables, getURL, loadMovie et l'entrée LoadVars dans le *Guide de référence du langage ActionScript 2.0*.

Utilisation de la classe LoadVars

Si vous publiez vers Flash Player 6 ou une version plus récente et avez besoin de davantage de souplesse que ce qui est offert par `loadVariables()`, utilisez la classe LoadVars pour transférer les variables entre le fichier SWF et le serveur.

La classe LoadVars a été introduite dans Flash Player 6 pour mettre en place une interface plus nette et orientée objet pour les tâches ordinaires d'échange de données CGI avec un serveur Web. Les avantages de la classe LoadVars comprennent les éléments suivants :

- Il n'est pas nécessaire de créer des clips conteneur pour conserver les données ni d'encombrer les clips existants avec des variables propres aux communications client/serveur.
- L'interface de classe est similaire à l'objet XML, ce qui permet de renforcer la cohérence du code ActionScript. Elle utilise les méthodes `load()`, `send()` et `sendAndLoad()` pour lancer les communications avec un serveur. La principale différence entre les classes LoadVars et XML réside dans le fait que les données LoadVars sont une propriété de l'objet LoadVars, et non une arborescence DOM (Document Object Model) XML stockée dans l'objet XML.
- L'interface de classe est plus simple, avec des méthodes appelées `load`, `send`, `sendAndLoad`, que l'ancienne interface loadVariables.

- Vous trouverez davantage d'informations sur les communications en utilisant les méthodes `getBytesLoaded` et `getBytesTotal`.
- Vous pouvez obtenir des informations sur la progression du téléchargement de vos données (bien que vous ne puissiez pas accéder aux données avant la fin de leur téléchargement).
- L'interface de rappel se fait par l'intermédiaire des méthodes ActionScript (`onLoad`), ce qui remplace l'approche `onClipEvent` (données), obsolète et déconseillée, requise pour loadVariables.
- Vous disposez de notifications d'erreur.
- Vous pouvez ajouter des en-têtes de requête HTTP.

Vous devez créer un objet LoadVars pour appeler ses méthodes. Cet objet est un conteneur qui stocke les données chargées.

La procédure suivante montre comment utiliser ColdFusion et la classe LoadVars pour envoyer un message électronique à partir d'un fichier SWF.

REMARQUE	Vous devez disposer de ColdFusion sur votre serveur Web pour cet exemple.

Pour charger des données avec l'objet LoadVars :

1. Créez un fichier CFM dans Macromedia Dreamweaver ou dans votre éditeur de texte préféré. Ajoutez le texte suivant au fichier :

```
<cfif StructKeyExists(Form, "emailTo")>
<cfmail to="#Form.emailTo#" from="#Form.emailFrom#"
  subject="#Form.emailSubject#">#Form.emailBody#</cfmail>
&result=true
<cfelse>
&result=false
</cfif>
```

2. Enregistrez le fichier sous le nom **email.cfm** et transférez-le sur votre site Web.
3. Dans Flash, créez un document.
4. Créez quatre champs de saisie de texte sur la scène, et donnez-leur les noms d'occurrence suivants : **emailFrom_txt**, **emailTo_txt**, **emailSubject_txt**, et **emailBody_txt**.
5. Créez un champ de texte dynamique sur la scène avec le nom d'occurrence **debug_txt**.
6. Créez un symbole de bouton, faites glisser une occurrence sur la scène, puis nommez-la **submit_btn**.

7. Sélectionnez l'image 1 dans le scénario et ouvrez le panneau Actions (Fenêtre > Actions) si ce n'est pas déjà fait.
8. Tapez le code suivant dans le panneau Actions :

```
this.submit_btn.onRelease = function() {
  var emailResponse:LoadVars = new LoadVars();
  emailResponse.onLoad = function(success:Boolean) {
  if (success) {
       debug_txt.text = this.result;
  } else {
       debug_txt.text = "erreur lors du téléchargement du contenu";
  }
  };
  var email:LoadVars = new LoadVars();
  email.emailFrom = emailFrom_txt.text;
  email.emailTo = emailTo_txt.text;
  email.emailSubject = emailSubject_txt.text;
  email.emailBody = emailBody_txt.text;
  email.sendAndLoad("http://www.votresite.com/email.cfm",
  emailResponse, "POST");
};
```

   Ce code ActionScript crée une nouvelle occurrence d'objet LoadVars, copie les valeurs des champs texte dans l'occurrence, puis envoie les données au serveur. Le fichier CFM envoie le message électronique et renvoie une variable (`true` ou `false`) au fichier SWF appelé `result`, qui s'affiche dans le champ de texte `debug_txt`.

REMARQUE	N'oubliez pas de modifier l'URL www.yoursite.com sur votre propre domaine.

9. Enregistrez le document sous **sendEmail.fla**, puis publiez-le en sélectionnant Fichier > Publier.
10. Transférez sendEmail.swf vers le répertoire qui contient email.cfm (le fichier ColdFusion que vous avez enregistré et transféré à l'étape 2).
11. Affichez et testez le fichier SWF dans un navigateur.

Pour plus d'informations, consultez l'entrée LoadVars dans le *Guide de référence du langage ActionScript 2.0*.

Flash Player 8 et versions ultérieures prennent en charge le gestionnnaire d'événement `onHTTPStatus` pour les classes LoadVars class, XML et MovieClipLoader afin de permettre aux utilisateurs d'accéder au code de statut à partir d'une requête HTTP. Ceci permet aux développeurs de déterminer *pourquoi* une opération de chargement particulière peut échouer au lieu de pouvoir déterminer que l'opération de chargement a déjà échoué.

L'exemple suivant montre comment vous pouvez utiliser le gestionnaire d'événement `onHTTPStatus` de la classe LoadVars pour vérifier si un fichier texte a été téléchargé avec succès à partir d'un serveur et ce qu'était le code de statut retourné à partir de la requête HTTP.

Pour vérifier le statut HTTP avec l'objet LoadVars :

1. Créez un document Flash et enregistrez-le sous le nom **loadvars.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
this.createTextField("params_txt", 10, 10, 10, 100, 21);
params_txt.autoSize = "left";

var my_lv:LoadVars = new LoadVars();
my_lv.onHTTPStatus = function(httpStatus:Number) {
  trace("HTTP status is: " + httpStatus);
};
my_lv.onLoad = function(success:Boolean) {
  if (success) {
    trace("text file successfully loaded");
    params_txt.text = my_lv.dayNames;
  } else {
    params_txt.text = "unable to load text file";
  }
};
my_lv.load("http://www.helpexamples.com/flash/404.txt");
/* résultat :
  Erreur lors de l'ouverture de l'URL "http://www.helpexamples.com/
  flash/404.txt"
  l'état HTTP est : 404
*/
```

Le code précédent crée un nouveau champ de texte sur la scène et active le dimensionnement de champ de texte automatique. Ensuite, un objet LoadVars est créé et deux gestionnaires d'événement : `onHTTPStatus` et `onLoad`. Le gestionnaire d'événement `onHTTPStatus` (pris en charge dans Flash Player 8 et versions ultérieures) est invoqué lorsqu'une opération `LoadVars.load()` ou `LoadVars.sendAndLoad()` est complétée. La valeur passée à la fonction de gestionnaire d'événement `onHTTPStatus` (`httpStatus` dans le code précédent) contient la définition de code de statut HTTP pour l'opération de chargement actuelle. Si le fichier SWF a pu charger avec succès le fichier de texte, la valeur de `httpStatus` est définie sur 200 (code de statut HTTP pour « OK »). Si le fichier n'existait pas sur le serveur, la valeur de `httpStatus` est définie sur 404 (code de statut HTTP pour « Introuvable »). Le second gestionnaire d'événement `LoadVars.onLoad()` est appelé après que le fichier a fini son chargement. Si le chargement du fichier est réussi, la valeur du paramètre `success` est définie sur `true`, autrement le paramètre`success` est défini sur `false`. Finalement, le fichier externe est chargé en utilisant la méthode `LoadVars.load()`.

3. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

   Flash affiche un message d'erreur au panneau de sortie déclarant qu'il n'a pas pu charger l'image parce qu'elle n'existe pas sur le serveur. Le gestionnaire d'événement `onHTTPStatus` suit le code de statut 404 puisque le fichier n'a pas pu être touvé sur le serveur, et gestionnaire d'événement `onLoad` défini la propriété texte du champ de texte champ `params_txt` sur « impossible de charger le fichier texte.

ATTENTION	Si un serveur Web ne renvoie pas un code de statut à Flash Player, le chiffre 0 est retourné au gestionnaire d'événement `onHTTPStatus`.

Téléchargement de fichier depuis et vers le serveur

La classe FileReference permet de transférer et télécharger des fichiers entre un client et le serveur. Vos utilisateurs peuvent transférer et télécharger des fichiers entre leur ordinateur et un serveur. Les utilisateurs sont invités à sélectionner un fichier pour le charger ou un emplacement pour le télécharger dans une boîte de dialogue (telle que la boîte de dialogue Ouvrir dans le système d'exploitation Windows).

Chaque objet FileReference que vous créez avec ActionScript se rapporte à un seul fichier sur le disque dur de l'utilisateur. L'objet a des propriétés qui contiennent les informations à propos de la taille, du type, du nom, des dates de création et de modification du fichier. Sous Macintosh, il y a aussi une propriété pour le type de créateur du fichier.

Vous pouvez créer une occurrence de la classe FileReference de deux façons. Vous pouvez utiliser le `nouvel`opérateur suivant :

```
import flash.net.FileReference;
var myFileReference:FileReference = new FileReference();
```

Ou vous pouvez appeler la méthode `FileReferenceList.browse()`, qui ouvre une boîte de dialogue sur le système de l'utilisateur pour inviter celui-ci à sélectionner un fichier à transférer et puis crée un tableau d'objets FileReference si l'utilisateur sélectionne un ou plusieurs fichiers avec succès. Chaque objet FileReference représente un fichier sélectionné par l'utilisateur à partir de la boîte de dialogue. Un objet FileReference ne contient pas de données dans les propriétés FileReference (telle que `name`, `size`, ou`modificationDate`) jusqu'à ce que la méthode `FileReference.browse()` ou `FileReferenceList.browse()` ait été appelée et que l'utilisateur ait sélectionné un fichier à partir du sélecteur de fichiers ou que la méthode `FileReference.download()` ait été utilisée pour sélectionner un fichier à partir du sélecteur de fichiers.

REMARQUE

`FileReference.browse()` permet à l'utilisateur de sélectionner un seul fichier.
`FileReferenceList.browse()` permet à l'utilisateur de sélectionner plusieurs fichiers.

Lorsque l'appel de la méthode `browse()` a réussi, appelez `FileReference.upload()` pour charger un fichier à la fois.

Vous pouvez également ajouter une fonction de téléchargement à votre application Flash. La méthode `FileReference.download()` invite les utilisateurs finaux à sélectionner un emplacement sur leurs disques durs pour enregistrer le fichier à partir d'un serveur. Cette méthode lance aussi le téléchargement à partir d'une URL à distance. Lors de l'utilisation de `la méthode download()` seul le nom `FileReference.name` property est accessible quand l'événement `onSelect` est envoyé. Le reste des propriétés n'est pas accessible jusqu'à ce que l'événement `onComplete` soit envoyé.

REMARQUE

Lorsqu'une boîte de dialogue apparaît sur l'ordinateur de l'utilisateur final, l'emplacement par défaut qui apparaît dans la boîte de dialogue est le dernier dossier parcouru (si cet emplacement peut être déterminé) ou le bureau (s'il ne peut pas être déterminé). Les API FileReference et FileReferenceList ne vous permettent pas de définir l'emplacement du fichier par défaut.

Pour plus d'informations sur la fonctionnalité et la sécurité de l'API FileReference, consultez la section « A propos de la fonctionnalité et la sécurité de l'API FileReference », à la page 644. Pour un exemple d'une application qui utilise l'API FileReference, consultez la section « Ajout d'une fonctionnalité de chargement de fichier côté serveur à une application », à la page 646. Pour un exemple de fichier source, FileUpload.fla, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/FileUpload afin d'accéder à cet exemple.

Pour plus d'information sur chaque méthode, propriété et événement de l'API FileReference, consultez `FileReference (flash.net.FileReference)` et `FileReferenceList (flash.net.FileReferenceList)`dans le *Guide de référence du langage ActionScript 2.0*.

A propos de la fonctionnalité et la sécurité de l'API FileReference

Flash Player et l'API FileReference (voir « Téléchargement de fichier depuis et vers le serveur », à la page 643) prennent en charge le téléchargement de fichiers depuis et vers le serveur. Le lecteur n'impose pas de restrictions au niveau de la taille des fichiers chargés ou téléchargés, mais officiellement il prend en charge des chargements/téléchargements de 100 Mo au maximum. L'API FileReference *ne permet pas* à l'application Flash qui lance le transfert de fichiers d'effectuer ce qui suit :

- Accéder au fichier chargé ou téléchargé
- Accéder au chemin du fichier sur l'ordinateur de l'utilisateur.

Lorsqu'un serveur nécessite une authentification, la seule opération susceptible de réussir consiste à télécharger un fichier avec le module externe du navigateur Flash Player. Le chargement sur tous les lecteurs Flash, ou le téléchargement via un lecteur autonome ou externe, échoue sur un serveur nécessitant une authentification. Utilisez les écouteurs d'événements FileReference afin de déterminer si les opérations ont réussi, ou non, et pour traiter les erreurs.

Le chargement et le téléchargement de fichiers sont limités au domaine du fichier SWF, y compris les domaines spécifiés par un fichier de régulation inter-domaines. Vous devez placer un fichier de régulation sur le serveur si le fichier SWF qui lance le chargement ou téléchargement ne provient pas du même domaine que le serveur. Pour plus d'informations sur la sécurité et les fichiers de stratégie inter-domaines, consultez la section « Restriction des API de réseau », à la page 695.

Lorsque des appels à `FileReference.browse()`, `FileReferenceList.browse()`, ou `FileReference.download()` sont en cours d'exécution, la lecture du fichier SWF s'arrête sur les plates-formes suivantes : Les modules externes du navigateur Flash Player dans Mac OS X, le Flash Player externe de Macintosh, et le lecteur autonome de Macintosh sur Mac OS X 10.1 et antérieur. Le fichier SWF continue à s'exécuter sur tous les lecteurs Windows et sur Flash Player autonome Macinctosh sur Mac OS X 10.2 et version ultérieure.

AVERTISSEMENT

Lorsque vous autorisez des utilisateurs à charger des fichiers vers un serveur, vous devez toujours être prudent pour vérifier le type de fichier avant d'enregistrer le fichier sur le disque dur. Par exemple, vous ne voudriez pas autoriser un utilisateur à charger un script du côté serveur qui pourrait être utilisé pour supprimer des dossiers ou fichiers sur le serveur. Si vous voulez uniquement autoriser les utilisateurs à charger un fichier image, assurez-vous que le script du côté serveur qui charge les fichiers vérifie que le fichier chargé est une image valide.

Pour un exemple d'une application qui utilise l'API FileReference, consultez la section « Ajout d'une fonctionnalité de chargement de fichier côté serveur à une application », à la page 646.

Ajout d'une fonctionnalité de chargement de fichier côté serveur à une application

La procédure suivante montre comment construire une application qui vous permet de charger des fichiers images vers un serveur. L'application permet aux utilisateurs de sélectionner une image à charger sur leurs disques durs et puis de l'envoyer vers un serveur. L'image qu'ils chargent apparaît alors dans le fichier SWF qu'ils ont utilisé pour charger l'image.

Après l'exemple qui construit l'application Flash se trouve un exemple qui détaille le code du côté serveur. Rappelez-vous que les fichiers image sont limités en taille. Vous pouvez uniquement charger des images inférieures ou égales à 200 K.

Pour construire une application Flash avec l'API FileReference :

1. Créez un document Flash, puis enregistrez-le sous le nom de **fileref.fla**.
2. Ouvrez le panneau Composants, et puis faites glisser le composant ScrollPane sur la scène et donnez-lui un nom d'occurrence **imagePane**. (L'occurrence ScrollPane est dimensionnée et repositionnée avec ActionScript dans une étape ultérieure.)
3. Faites glisser un composant Button sur la scène et donnez-lui un nom d'occurrence de **uploadBtn**.
4. Faites glisser deux composants Label sur la Scène et nommez leurs occurrences **imageLbl** et **statusLbl**.
5. Faites glisser un composant ComboBox sur la scène et nommez son occurrence **imagesCb**.
6. Faites glisser un composant TextArea sur la scène et nommez son occurrence **statusArea**.
7. Créez un symbole de clip sur la scène et ouvrez le symbole pour l'édition (double-cliquez sur l'occurrence pour l'ouvrir en mode d'édition de symbole).
8. Créez un champ de texte statique dans le clip, et puis ajoutez le texte suivant :

   Le fichier que vous avez tenté de télécharger n'est pas sur le serveur.

   Dans l'application finale, cet avertissement peut apparaître pour l'une des raisons suivantes, parmi d'autres :

   - L'image a été supprimée de la liste d'attente sur le serveur tandis que d'autres images étaient chargées.
   - Le serveur n'a pas copié l'image parce que sa taille dépassait 200K.
   - Le type de fichier n'était pas un fichier JPEG, GIF, ou PNG valide.

REMARQUE	La largeur du champ de texte doit être inférieure à celle de l'occurrence ScrollPane (400 pixels) ; autrement les utilisateurs doivent faire défiler horizontalement pour voir le message d'erreur.

9. Cliquez avec le bouton droit de la souris sur le symbole dans la Bibliothèque clip et choisissez Liaison dans le menu contextuel.
10. Sélectionnez Exporter pour ActionScript, et Exporter dans la case à cocher de la première image, et tapez **Message** dans la zone de texte Identifiant. Cliquez sur OK.
11. Ajoutez l'ActionScript suivant à l'Image 1 du scénario :

REMARQUE	Les commentaires de code comprennent les détails à propos de la fonctionnalité. Une présentation générale du code suit cet exemple.

```
import flash.net.FileReference;

imagePane.setSize(400, 350);
imagePane.move(75, 25);
uploadBtn.move(75, 390);
uploadBtn.label = "Upload Image";
imageLbl.move(75, 430);
imageLbl.text = "Select Image";
statusLbl.move(210, 390);
statusLbl.text = "Status";
imagesCb.move(75, 450);
statusArea.setSize(250, 100);
statusArea.move(210, 410);

/* L'objet écouteur est à l'écoute des événements FileReference. */
var listener:Object = new Object();

/* Lorsque l'utilisateur sélectionne un fichier, la méthode onSelect()
  est appelée, et a passé une référence à l'objet FileReference. */
listener.onSelect = function(selectedFile:FileReference):Void {
  /* Mettre à jour TextArea pour notifier l'utiliateur que Flash tente
  de charger l'image. */
  statusArea.text += "Attempting to upload " + selectedFile.name + "\n";
  /* Charger le fichier vers le script PHP sur le serveur. */
  selectedFile.upload("http://www.helpexamples.com/flash/file_io/
  uploadFile.php");
};

/*Lorsque le fichier commence à se charger, la méthode onSelect() est
  appelée, alors aviser l'utilisateur que le fichier commence son
  chargement. */
listener.onOpen = function(selectedFile:FileReference):Void {
  statusArea.text += "Opening " + selectedFile.name + "\n";
};
```

```
/* Lorsque le fichier est chargé, la méthode onComplete() est appelée. */
listener.onComplete = function(selectedFile:FileReference):Void {
  /* Notifier l'utiliateur que Flash commence à télécharger l'image. */
  statusArea.text += "Downloading " + selectedFile.name + " to
  player\n";
  /* Ajouter l'image au composant ComboBox. */
  imagesCb.addItem(selectedFile.name);
  /* Définir l'index sélectionné du ComboBox sur celui de l'image
  ajoutée le plus récemment. */
  imagesCb.selectedIndex = imagesCb.length - 1;
  /* Appeller la fonction personnalisée downloadImage(). */
  downloadImage();
};

var imageFile:FileReference = new FileReference();
imageFile.addListener(listener);

imagePane.addEventListener("complete", imageDownloaded);
imagesCb.addEventListener("change", downloadImage);
uploadBtn.addEventListener("click", uploadImage);

/* Si l'image ne se télécharge pas, la propriété totale de l'objet
  d'événement sera égale à -1. Dans ce cas, afficher un message à
  l'utilisateur. */
function imageDownloaded(event:Object):Void {
  if (event.total == -1) {
    imagePane.contentPath = "Message";
  }
}

/* Lorsqée l'utilisateur sélectionne une image  partir du ComboBox, ou
  lorsque la fonction downloadImage() est appelée directement à partir
  de la méthode listener.onComplete(), la fonction downloadImage()
  définit le contentPath du ScrollPane afin de commencer le
  téléchargement de l'image dans le lecteur. */
function downloadImage(event:Object):Void {
  imagePane.contentPath = "http://www.helpexamples.com/flash/file_io/
  images/" + imagesCb.value;
}

Lorsque l'utilisateur clique sur le bouton, Flash appelle la fonction
  uploadImage() et il ouvre une boîte de dialogue de navigateur de
  fichiers. */
function uploadImage(event:Object):Void {
  imageFile.browse([{description: "Image Files", extension:
  "*.jpg;*.gif;*.png"}]);
}
```

Le code ActionScript importe d'abord la classe File Reference et initialise, positionne et redimensionne chacun des composants sur la scène. Ensuite, un objet écouteur est défini, et trois gestionnaires d'événement sont définis : `onSelect`, `onOpen`, et `onComplete`. L'objet écouteur est ensuite ajouté à un nouvel objet FileReference nommé `imageFile`. Ensuite, les écouteurs d'événement sont ajoutés à l'occurrence `imagePane` de ScrollPane, à l'occurrence `imagesCb` de ComboBox et à l'occurrence `uploadBtn` de Button. Chacun des écouteurs d'événement est défini dans le code qui suit cette section de code.

La première fonction, `imageDownloaded()`, vérifie si la quantité d'octets totaux pour les images téléchargées est de -1, et dans ce cas, elle définit le`contentPath` pour l'occurrence ScrollPane sur le clip avec l'identifiant de liaison de Message que vous avez créé dans une étape précédente. La seconde fonction, `downloadImage()`, tente de télécharger l'image récemment chargée dans l'occurrence ScrollPane. Lorsque l'image a été téléchargée, la fonction `imageDownloaded()` définie précédemment est enclenchée et vérifie si l'image a été téléchargée avec succès. La fonction finals, `uploadImage()`, ouvre une boîte de dialogue de navigateur de fichiers qui filtre toutes les images JPEG, GIF et PNG.

12. Enregistrez les modifications apportées au document.
13. Sélectionnez Fichier > Paramètres de publication, puis sélectionnez l'onglet Formats et assurez-vous que Flash et HTML sont tous les deux sélectionnés.
14. (Optionnel) Dans la boîte de dialogue Paramètres de publication, sélectionnez l'onglet Flash, et puis sélectionnez Accès au réseau uniquement à partir du menu contextuel Sécurité de lecture locale.

    Si vous complétez cette étape, vous ne rencontrerez pas de restrictions de sécurité si vous testez votre document dans un navigateur local.
15. Dans la boîte de dialogue Paramètres de publication, cliquez sur Publier pour Créer les fichiers HTML et SWF.

    Quand c'est fait, passez à la procédure suivante dans laquelle vous créez le conteneur pour le fichier SWF.

Pour un exemple de fichier source, FileUpload.fla, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/FileUpload afin d'accéder à cet exemple.

La procédure suivante nécessite que PHP soit installé sur votre serveur Web et que vous ayez les autorisations d'écriture sur les images nommées de sous-dossiers et les temporaires. Vous devez d'abord terminer la procédure précédente, ou utiliser le fichier SWF terminé disponible dans les dossiers notés précédemment.

Pour créer un script du côté serveur pour l'application de chargement d'images :

1. Créez un document PHP dans un éditeur tel que Dreamweaver ou Notepad.
2. Ajoutez le code PHP suivant au document. (Une présentation générale du code suit ce script.)

```
<?php

$MAXIMUM_FILESIZE = 1024 * 200; // 200 Ko
$MAXIMUM_FILE_COUNT = 10; // conserver au maximum 10 fichiers sur
                          // le serveur
echo exif_imagetype($_FILES['Filedata']);
if ($_FILES['Filedata']['size'] <= $MAXIMUM_FILESIZE) {
  move_uploaded_file($_FILES['Filedata']['tmp_name'], "./temporary/
  ".$_FILES['Filedata']['name']);
  $type = exif_imagetype("./temporary/".$_FILES['Filedata']['name']);
  if ($type == 1 || $type == 2 || $type == 3) {
    rename("./temporary/".$_FILES['Filedata']['name'], "./images/
  ".$_FILES['Filedata']['name']);
  } else {
    unlink("./temporary/".$_FILES['Filedata']['name']);
  }
}
$directory = opendir('./images/');
$files = array();
while ($file = readdir($directory)) {
  array_push($files, array('./images/'.$file, filectime('./images/
  '.$file)));
}
usort($files, sorter);
if (count($files) > $MAXIMUM_FILE_COUNT) {
  $files_to_delete = array_splice($files, 0, count($files) -
  $MAXIMUM_FILE_COUNT);
  for ($i = 0; $i < count($files_to_delete); $i++) {
    unlink($files_to_delete[$i][0]);
  }
}
print_r($files);
closedir($directory);

function sorter($a, $b) {
  if ($a[1] == $b[1]) {
    return 0;
  } else {
    return ($a[1] < $b[1]) ? -1 : 1;
  }
}
?>
```

Ce code PHP définit d'abord deux variables constantes : `$MAXIMUM_FILESIZE` et `$MAXIMUM_FILE_COUNT`. Ces variables dictent la taille maximum (en kilooctets) d'une image étant chargée sur le serveur (200Ko), ainsi que beaucoup de fichiers chargés récemment peuvent être conservés dans le dossier d'images (10). Si la taille du fichier de l'image actuellement en chargement est inférieure ou égale à la valeur de `$MAXIMUM_FILESIZE`, l'image est déplacée dans un dossier temporaire.

Ensuite, le type de fichier de l'image chargée est vérifié pour assurer que l'image est un fichier JPEG, GIF, ou PNG. Si l'image est un type d'image compatible, elle est copiée du dossier temporaire vers le dossier d'images. Si l'image chargée n'est pas un des types autorisés, elle est supprimée du système de fichiers.

Ensuite, une liste de répertoires du dossier d'images est créée et passée en boucle avec la boucle while. Chaque fichier dans le dossier d'images est ajouté à un tableau et puis trié. Si le nombre actuel de fichiers dans le dossier d'images est supérieur à la valeur de `$MAXIMUM_FILE_COUNT`, les fichiers sont supprimés jusqu'à qu'il y ait uniquement `$MAXIMUM_FILE_COUNT` images restantes. Ceci empêche le dossier d'images de croître à une taille ingérable, car il ne peut y avoir que 10 images dans le dossier à tout moment, et chaque image peut uniquement être de 200Ko ou moins (ou environ 2 Mo d'images à tout moment).

3. Enregistrez les modifications apportées au document PHP.
4. Chargez les fichiers SWF, HTML et PHP vers votre serveur Web.
5. Visualisez le document HTML distant dans le navigateur Web, et cliquez sur le bouton Charger image dans le fichier SWF.
6. Recherchez un fichier image sur votre disque dur et sélectonnez Ouvrir dans la boîte de dialogue.

   Le fichier SWF charge le fichier image vers le document PHP à distance, et l'affiche dans le ScrollPane (qui ajoute des barres de défilement si nécessaire). Si vous voulez visualiser une image précédemment chargée, vous pouvez sélectionner le nom du fichier à partir de l'occurence ComboBox sur la scène. Si l'utilisateur tente de charger une image qui n'est pas d'un type autorisé (uniquement une image JPEG, GIF, ou PNG est autorisée) ou la taille du fichier est trop grande (plus de 200 Ko), Flash affiche le message d'erreur provenant du clip Message dans la Bibliothèque.

Pour un exemple de fichier source, FileUpload.fla, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/FileUpload afin d'accéder à cet exemple.

Pour plus d'informations sur la sécurité de fichiers locaux, consultez « Sécurité des fichiers locaux et Flash Player », à la page 677.

Pour plus d'informations sur la rédaction de fichiers PHP, consultez le site www.php.net/.

Présentation du langage XML

Le langage XML (Extensible Markup Language) est en passe de devenir la norme d'échange de données structurées dans les applications Internet. Vous pouvez intégrer les données de Flash avec des serveurs qui utilisent la technologie XML pour construire des applications sophistiquées telles que des services de discussion en ligne ou de courtage.

En XML, tout comme en HTML, vous utilisez des balises pour *marquer*, ou définir, une chaîne de texte. Dans le langage HTML, vous utilisez des balises prédéfinies pour indiquer la façon dont le texte doit apparaître dans un navigateur Web (par exemple, la balise `<b>` indique que le texte doit être en gras). Dans le langage XML, vous définissez des balises qui identifient le type d'une partie de données (par exemple, `<password>VerySecret</password>`). Le langage XML sépare la structure des informations de leur mode d'affichage, ce qui permet d'utiliser un même document XML dans des environnements différents.

Chaque balise XML est appelée *nœud* ou élément. Chaque nœud possède un type (1, qui indique un élément XML ou 3, qui indique un nœud texte) et les éléments peuvent également posséder des attributs. Un nœud imbriqué dans un autre est appelé *nœud enfant*. Cette structure hiérarchique de nœuds est appelée DOM XML, un peu comme le DOM JavaScript, qui correspond à la structure des éléments dans un navigateur Web.

Dans l'exemple suivant, `<portfolio>` est le nœud parent ; il ne comporte pas d'attributs et contient le nœud enfant `<holding>`, qui a les attributs `symbol`, `qty`, `price` et `value` :

```
<portfolio>
  <holding symbol="rich"
    qty="75"
    price="245.50"
    value="18412.50" />
</portfolio>
```

Pour plus d'informations, consultez les sections suivantes :

- « Utilisation de la classe XML », à la page 653
- « Utilisation de la classe XMLSocket », à la page 659

Pour plus d'informations sur XML, consultez www.w3.org/XML.

Pour un exemple de fichier source, xml_blogTracker.fla, qui décrit comment créer un traqueur de journaux du Web en chargeant, en analysant et en manipulant des données XML, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/XML_BlogTracker afin d'accéder à cet exemple.

Pour un exemple de fichier source, xml_languagePicker.fla, qui décrit comment utiliser le langage XML et des tableaux imbriqués pour sélectionner des chaînes de langages différents afin de remplir les champs de texte, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/XML_LanguagePicker afin d'accéder à cet exemple.

Pour un exemple de fichier source, xmlmenu.fla, qui décrit comment créer un menu dynamique avec des données XML, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/XML_Menu afin d'accéder à cet exemple. Cet exemple appelle le constructeur ActionScript `XmlMenu()` et lui transmet deux paramètres : le chemin au fichier menu XML et une référence au scénario. Le reste de la fonctionnalité réside dans un fichier de classe personnalisée, XmlMenu.as.

Utilisation de la classe XML

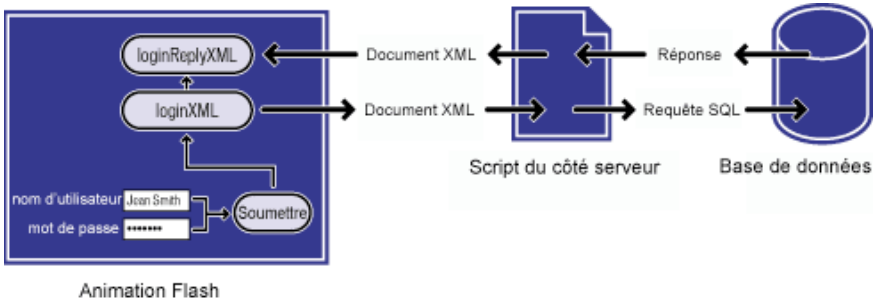
Les méthodes de la classe XML ActionScript (par exemple, `appendChild()`, `removeNode()` et `insertBefore()`) permettent de structurer les données XML dans Flash qui doivent être envoyées à un serveur et de manipuler et interpréter les données XML téléchargées.

Les méthodes de la classe XML suivantes permettent d'échanger des données XML avec un serveur à l'aide de la méthode HTTP `POST` :

- La méthode `load()` télécharge le code XML depuis une URL et le place dans un objet XML ActionScript.
- La méthode `send()` code l'objet XML dans le document XML et l'envoie à une URL spécifiée en utilisant la méthode `POST`. Si spécifié, les résultats s'affichent dans une fenêtre de navigateur.
- La méthode `sendAndLoad()` envoie un objet XML à une URL. Toutes les informations renvoyées sont placées dans un objet XML ActionScript.

Par exemple, vous pourriez créer un système de courtage qui stockerait toutes ses informations (noms d'utilisateur, mots de passe, identifiants de session, contenu des portefeuilles et informations de transaction) dans une base de données.

Le script côté serveur qui transmet les informations entre Flash et la base de données lit et écrit les données au format XML. Vous pouvez utiliser ActionScript pour convertir les informations récupérées dans le fichier SWF (par exemple, un nom d'utilisateur et un mot de passe) en un objet XML et envoyer ensuite les données au script côté serveur sous forme de document XML. Vous pouvez également utiliser ActionScript pour charger le document XML que le serveur renvoie dans un objet XML à utiliser dans le fichier SWF.



Flux et conversion des données entre un fichier SWF, un script côté serveur et une base de données

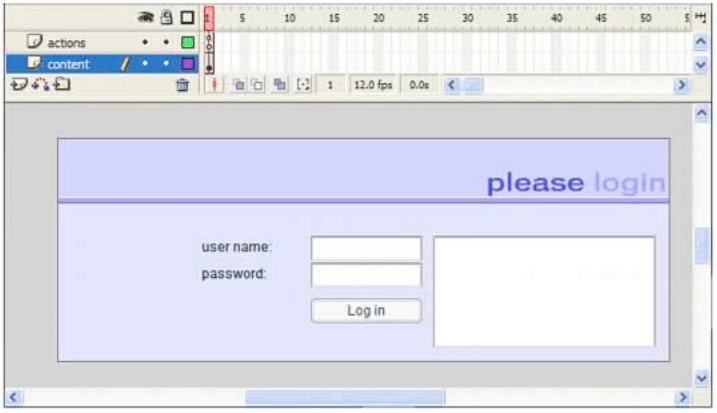
La validation du mot de passe pour le système de courtage nécessite deux scripts : une fonction définie dans l'image 1 et un script qui crée, puis envoie les objets XML créés dans le document.

Lorsqu'un utilisateur entre des informations dans les champs de texte du fichier SWF avec les variables `username` et `password`, les variables doivent être converties au format XML avant d'être transmises au serveur. La première section du script charge les variables dans un objet XML nouvellement créé et appelé `loginXML`. Quand un utilisateur clique sur un bouton pour se connecter, l'objet `loginXML` est converti en chaîne XML et envoyé au serveur.

Le code ActionScript suivant est placé sur le scénario et est utilisé pou envoyer les données formatées XML au serveur. Pour comprendre le script, vous pourrez vous aider des commentaires (indiqués par les caractères `//`) :

```
//ignorer les espaces blancs XML
XML.prototype.ignoreWhite = true;
// Construire un objet XML contenant la réponse du serveur
var loginReplyXML:XML = nouveau XML();
// cette fonction se déclenche quand un paquet XML est reçu du serveur.
loginReplyXML.onLoad = function(success:Boolean) {
  if (success) {
    //(facultatif) Crée deux champs texte pour l'état/débogage
    //status_txt.text = this.firstChild.attributes.status;
    //debug_txt.text = this.firstChild;
    switch (this.firstChild.attributes.STATUS) {
```

```
      case 'OK' :
        _global.session = this.firstChild.attributes.SESSION;
        trace(_global.session);
        gotoAndStop("welcome");
        break;
      case 'FAILURE' :
        gotoAndStop("loginfailure");
        break;
      default :
        // ceci ne devrait jamais arriver
        trace("Unexpected value received for STATUS.");
      }
    } else {
      trace("an error occurred.");
    }
};
// cette fonction se déclenche lorsque la connexion login_btn est cliquée
target_mc.onRelease = function() {
  var loginXML:XML = new XML();
  //crée des données au format XML à envoyer au serveur
  var loginElement:XMLNode = loginXML.createElement("login");
  loginElement.attributes.username = username_txt.text;
  loginElement.attributes.password = password_txt.text;
  loginXML.appendChild(loginElement);
  // Envoie les données mises au format XML au serveur
  loginXML.sendAndLoad("http://www.flash-mx.com/mm/main.cfm",
  loginReplyXML);
};
```



Vous pouvez tester ce code avec le nom d'utilisateur `JeanSmith` et le mot de passe `VerySecret`. La première section du script génère le code XML suivant lorsque l'utilisateur clique sur le bouton de connexion :

```
<login username="JeanSmith" password="VerySecret" />
```

Le serveur reçoit le code XML, génère une réponse XML et la renvoie au fichier SWF. Si le mot de passe est accepté, le serveur envoie la réponse suivante :

```
<LOGINREPLY STATUS="OK" SESSION="4D968511" />
```

Ce XML comprend un attribut `session` qui contient un ID unique, de session générée au hasard, qui est utilisé dans toutes les communications entre le client et le serveur pour le reste de la session. Si le mot de passe est rejeté, le serveur répond par le message suivant :

```
<LOGINREPLY STATUS="FAILURE" />
```

Le noeud XML`loginreply` doit charger dans un objet XML vide dans le fichier SWF. L'instruction suivante crée l'objet XML `loginreplyXML` pour recevoir le nœud XML :

```
// Construire un objet XML contenant la réponse du serveur
var loginReplyXML:XML = nouveau XML();
loginReplyXML.onLoad = function(success:Boolean) {
```

La deuxième déclaration dans ce code ActionScript définit une fonction anonyme (inline), qui est appelée quand l'événement `onLoad` se déclenche.

Le bouton de connexion (occurrence `login_btn`) permet d'envoyer le nom d'utilisateur et le mot de passe au format XML au serveur et de charger la réponse XML dans le fichier SWF. Vous pouvez utiliser la méthode `sendAndLoad()` pour ce faire, comme indiqué dans l'exemple suivant :

```
loginXML.sendAndLoad("http://www.flash-mx.com/mm/main.cfm", loginReplyXML);
```

Tout d'abord, les données au format XML sont créées, en utilisant les valeurs que l'utilisateur entre dans le fichier SWF. Cet objet XML est ensuite envoyé avec la méthode `sendAndLoad`. Comme pour les données provenant de la fonction `loadVariables()`, l'élément XML `loginreply` arrive de façon asynchrone (sans attendre les résultats avant d'être renvoyé) et se charge dans l'objet `loginReplyXML`. Lorsque les données arrivent, le gestionnaire `onLoad` de l'objet `loginReplyXML` est appelé. Vous devez définir la fonction `loginReplyXML`, qui est appelée lorsque le gestionnaire `onLoad` se déclenche, de façon à pouvoir traiter l'élément `loginreply`.

REMARQUE	Cette fonction doit toujours figurer sur l'image qui contient le code ActionScript relatif au bouton connexion.

Si la connexion aboutit, le fichier SWF passe à l'étiquette d'image `welcome`. Si la connexion ne réussit pas, la tête de lecture se place sur l'étiquette d'image `loginfailure`. Ceci a été traité en utilisant une condition et une instruction case.

Le fichier CFM contient le code suivant :

```
<cfif (Compare(Form.username, "Herbert") EQ 0) AND (Compare(Form.password,
  "glasses") EQ 0)>
  <cfoutput>&isValidLogin=1&session=#URLEncodedFormat(CreateUUID())#</
  cfoutput>
  <cfelse>
  <cfoutput>&isValidLogin=0</cfoutput>
</cfif>
```

Pour plus d'informations sur les instructions case et break, consultez les sections instruction case et instruction break dans le *Guide de référence du langage ActionScript 2.0*. Pour plus d'informations sur les conditions, consultez les sections instruction if et instruction else dans le *Guide de référence du langage ActionScript 2.0*.

REMARQUE	Ce design est seulement un exemple, et Adobe ne veut rien dire au sujet du niveau de sécurité qu'il fournit. Si vous mettez en œuvre un système sécurisé protégé par mot de passe, assurez-vous de bien connaître les concepts de la sécurité réseau.

Pour plus d' informations, consultez Intégration de XML et Flash dans une application Web à l'adresse http://www.adobe.com/support/flash/applications/xml/ et l'entrée XML dans le *Guide de référence du langage ActionScript 2.0*. Pour plus d'informations sur la sécurité de fichiers locaux, consultez « Sécurité des fichiers locaux et Flash Player », à la page 677.

Pour un exemple de fichier source, login.fla, qui décrit comment ajouter une fonctionnalité de connexion à vos sites Web à l'aide d'ActionScript 2.0, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/Login afin d'accéder à cet exemple. Cet exemple utilise ActionScript et ses composants pour créer un petit formulaire dans lequel vous entrez un nom d'utilisateur et un mot de passe, puis cliquez un bouton pour entrer sur un site.

Flash Player 8 et versions ultérieures prennent en charge le gestionnnaire d'événement `onHTTPStatus` pour les classes XML, LoadVars et MovieClipLoader afin de permettre aux utilisateurs d'accéder au code de statut à partir d'une requête HTTP. Ceci permet aux développeurs de déterminer *pourquoi* une opération de chargement particulière peut échouer au lieu de pouvoir déterminer que l'opération de chargement a déjà échoué.

L'exemple suivant montre comment vous pouvez utiliser le gestionnaire d'événement `onHTTPStatus` de la classe XML pour vérifier si un fichier XML a été téléchargé avec succès à partir d'un serveur et ce qu'était le code de statut retourné à partir de la requête HTTP.

Vérification des codes d'état HTTP en utilisant la classe XML

1. Créez un document Flash, puis enregistrez-le sous le nom de **fileref.fla**.
2. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onHTTPStatus = function(httpStatus:Number) {
  trace("HTTP status is: " + httpStatus);
};
my_xml.onLoad = function(success:Boolean) {
  if (success) {
    trace("XML successfully loaded");
    // 0 (Pas d'erreur : l'analyse est terminée.)
    trace("XML status is: " + my_xml.status);
  } else {
    trace("unable to load XML");
  }
};
my_xml.load("http://www.helpexamples.com/crossdomain.xml");
```

   Le code précédent définit un nouvel objet XML avec le nom variable `my_xml`, définit deux gestionnaires d'événement (`onHTTPStatus` et `onLoad`) et charge un fichier externe XML. Le gestionnaire d'événement `onLoad` s'assure que le fichier XML a été chargé.
   Dans l'affirmative, il envoie un message au panneau Sortie, ce qui permet de suivre la propriété d'état de l'objet XML. Il est important de se rappeler que le gestionnaire d'événement `onHTTPStatus` renvoie le code d'état retourné du serveur web, tandis que la propriété `XML.status` contient une valeur numérique qui indique si l'objet XML a été analysé correctement.
3. Choisissez Contrôle > Tester l'animation pour tester le document Flash.

AVERTISSEMENT

Ne pas confondre les codes httpStatus HTTP avec la propriété `status` de la classe XML. Le gestionnaire d'événement `onHTTPStatus` renvoie le code d'état du serveur d'une requête HTTP et la propriété `status` définit et renvoie automatiquement une valeur numerique qui indique si le document XML a été transformé correctement en objet XML.

ATTENTION

Si un serveur Web ne renvoie pas un code `status` à Flash Player, le nombre 0 est renvoyé au gestionnaire d'événement `onHTTPStatus`.

Pour un exemple de fichier source, xml_blogTracker.fla, qui décrit comment créer un traqueur de journaux du Web en chargeant, en analysant et en manipulant des données XML, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/XML_BlogTracker afin d'accéder à cet exemple.

Pour un exemple de fichier source, xml_languagePicker.fla, qui décrit comment utiliser le langage XML et des tableaux imbriqués pour sélectionner des chaînes de langages différents afin de remplir les champs de texte, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/XML_LanguagePicker afin d'accéder à cet exemple.

Pour un exemple de fichier source, xmlmenu.fla, qui décrit comment créer un menu dynamique avec des données XML, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/XML_Menu afin d'accéder à cet exemple. Cet exemple appelle le constructeur ActionScript `XmlMenu()` et lui transmet deux paramètres : le chemin au fichier menu XML et une référence au scénario en cours. Le reste de la fonctionnalité réside dans un fichier de classe personnalisé, XmlMenu.as.

Utilisation de la classe XMLSocket

ActionScript fournit une classe XMLSocket intégrée qui vous permet d'établir une connexion continue avec un serveur. Une connexion socket permet au serveur de publier l'information sur le client ou de la *pousser* dès qu'elle est disponible. En l'absence de connexion continue, le serveur devra attendre une requête HTTP. Cette connexion ouverte supprime les périodes d'attente et est souvent utilisée dans des applications en temps réel telles que les dialogues en ligne. Les données sont envoyées sur la connexion socket sous forme d'une chaîne et doivent être au format XML. Vous pouvez utiliser la classe XML pour structurer les données.

Pour créer une connexion socket, vous devez créer une application côté serveur qui attendra la requête de connexion socket et enverra une réponse au fichier SWF. Ce type d'application côté serveur peut être écrit dans un langage tel que Java.

REMARQUE

La classe XMLSocket ne peut pas automatiquement tunneler à travers les pare-feux parce que, au contaire du Real-Time Messaging Protocol (RTMP), le XMLSocket n'a pas de capacité de tunneling HTTP. Si vous devez utiliser le tunneling HTTP, envisagez l'utilisation de Flash Remoting ou Flash Media Server (qui prend en charge RTMP).

Vous pouvez utiliser les méthodes `connect()` et `send()` de la classe XMLSocket pour transférer un objet XML vers et à partir d'un serveur sur une connexion socket. La méthode `connect()` établit une connexion socket avec le port d'un serveur Web. La méthode `send()` transmet un objet XML au serveur spécifié dans la connexion socket.

Lorsque vous invoquez la méthode `connect()`, Flash Player ouvre une connexion TCP/IP avec le serveur et garde cette connexion ouverte jusqu'à ce que l'un des événements suivants se produise :

- La méthode `close()` de la classe XMLSocket est appelée.
- Il n'existe plus aucune référence à l'objet XMLSocket.
- Flash Player se ferme.
- La connexion est interrompue (le modem est déconnecté, par exemple).

L'exemple suivant crée une connexion socket XML et envoie les données de l'objet XML `myXML`. Pour comprendre le script, vous pourrez vous aider des commentaires (indiqués par les caractères `//`) :

```
// Créer l'objet XMLSocket
var theSocket:XMLSocket = new XMLSocket();
// Se connecter à un site sur un port non utilisé au-dessus de 1024 en
// utilisant la methode connect().
// Entrer localhost ou 127.0.0.1 pour des essais locaux.
// Pour un serveur en direct, entrez votre nom de domaine www.yourdomain.com
theSocket.connect("localhost", 12345);
// affiche le texte relatif à la connexion
theSocket.onConnect = function(myStatus) {
  if (myStatus) {
    conn_txt.text = "connection successful";
  } else {
    conn_txt.text = "no connection made";
  }
};
//données à envoyer
function sendData() {
  var myXML:XML = new XML();
  var mySend = myXML.createElement("thenode");
  mySend.attributes.myData = "someData";
  myXML.appendChild(mySend);
  theSocket.send(myXML);
}
// le bouton envoie des données
sendButton.onRelease = function() {
  sendData();
};
// suit les données renvoyées par la connexion socket
theSocket.onData = function(msg:String):Void {
  trace(msg);
};
```

Pour plus d'informations, consultez l'entrée XMLSocket dans le *Guide de référence du langage ActionScript 2.0*.

Pour plus d'informations sur la sécurité de fichiers locaux, consultez « Sécurité des fichiers locaux et Flash Player », à la page 677.

Echange de messages avec Flash Player

Pour envoyer des messages d'un fichier SWF à un environnement hôte (par exemple, un navigateur Web, un film Directeur Macromedia, ou le Flash Player autonome), vous pouvez utiliser la fonction `fscommand()`. Cette fonction vous permet d'étendre votre fichier SWF en utilisant les capacités de l'hôte. Par exemple, vous pouvez transmettre une fonction `fscommand()` à une fonction JavaScript dans une page HTML qui ouvre une nouvelle fenêtre de navigateur avec des propriétés spécifiques.

Pour contrôler un fichier SWF dans Flash Player avec des langages de navigation Web, tels que JavaScript, VBScript et Microsoft JScript, vous pouvez utiliser les méthodes Flash Player (fonctions qui envoient des messages depuis un environnement hôte vers le fichier SWF). Par exemple, vous pouvez disposer d'un lien dans une page HTML qui envoie votre fichier SWF vers un cadre spécifique.

Pour plus d'informations, consultez les sections suivantes :

- « Utilisation de la fonction fscommand() », à la page 661
- « Utilisation de JavaScript pour contrôler les applications Flash », à la page 664
- « Présentation des méthodes Flash Player », à la page 665

Utilisation de la fonction fscommand()

REMARQUE

L'API externe vient remplacer `fscommand()` dans Flash 8 et versions ultérieures pour interopérer avec une HTML ou une application conteneur. L'API externe offre une fonctionalité plus robuste que `fscommand()` dans cette situation. Pour plus d'informations, voir « A propos de l'API externe », à la page 665.

Vous utilisez la fonction `fscommand()` pour envoyer un message à un programme quel qu'il soit hébergeant Flash Player, tel qu'un navigateur web.

REMARQUE

L'utilisation de `fscommand()` pour appeler JavaScript ne fonctionne pas avec les navigateurs Safari ou Internet Explorer pour le Macintosh.

La fonction `fscommand()` dispose de deux paramètres : *`command`* et *`arguments`*. Pour envoyer un message à la version autonome de Flash Player, vous devez utiliser des commandes et des arguments prédéfinis. Par exemple, le gestionnaire d'événement suivant définit le lecteur autonome pour qu'il affiche le fichier SWF en taille plein écran lorsque le bouton est relâché :

```
my_btn.onRelease = function() {
  fscommand("fullscreen", true);
};
```

Le tableau suivant indique les valeurs que vous pouvez spécifier pour les paramètres *`command`* et *`arguments`* de `fscommand()` pour contrôler la lecture et l'aspect d'un fichier SWF lu dans le lecteur autonome, y compris les projections.

REMARQUE

Une *projection* est un fichier SWF sauvegardé dans un format qui peut fonctionner comme une application autonome–qui incorpore Flash Player avec votre contenu dans un fichier exécutable.

Commande	Arguments	Rôle
`quit`	Aucun	Ferme la projection.
`fullscreen`	`true` ou `false`	La spécification de `true` définit Flash Player en mode plein écran. La spécification de `false` renvoie le lecteur en affichage normal du menu.
`allowscale`	`true` ou `false`	La spécification de `false` définit le lecteur de sorte que le fichier SWF soit toujours affiché dans sa taille originale et que son échelle ne soit jamais modifiée. La spécification de `true` oblige le fichier SWF à adopter l'échelle 100 % du lecteur.
`showmenu`	`true` ou `false`	La spécification de `true` active le jeu complet des éléments de menu contextuel. La spécification de `false` masque tous les éléments de menu contextuel, à l'exception de Paramètres et A propos de Flash Player.
`exec`	Chemin de l'application	Exécute une application depuis la projection.

Pour utiliser `fscommand()` pour envoyer un message à un langage de programmation tel que JavaScript dans un navigateur Web, vous pouvez transmettre deux arguments quelconques avec les paramètres *`command`* et *`arguments`*. Ces paramètres peuvent etre des chaînes ou expressions et sont utilisés dans une fonction JavaScript qui « attrape » ou manie la fonction `fscommand()`.

Une fonction `fscommand()` invoque la fonction JavaScript *moviename*`_DoFSCommand` dans la page HTML qui contient le fichier SWF, où *moviename* est le nom de Flash Player tel qu'il est affecté par l'attribut `name` de la balise `embed` ou par l'attribut `id` de la balise `object`. Si le fichier SWF a été affecté du nom `myMovie`, la fonction JavaScript invoquée est `myMovie_DoFSCommand`.

Pour utiliser `fscommand()` afin d'ouvrir une boîte de messages à partir d'un fichier SWF dans la page HTML à l'aide de JavaScript :

1. Créez un document FLA, puis enregistrez-le sous le nom de **myMovie.fla**.
2. Faites glisser deux occurrences du composant Button sur la Scène et donnez-leur les noms d'occurrence **window_btn** et **alert_btn**, respectivement, et les étiquettes **Open Window** et **Alert**.
3. Insérez un nouveau calque sur le scénario, et renommez le **Actions**.
4. Sélectionnez l'image 1 du calque Actions et, dans le panneau Actions, saisissez ce qui suit :

```
window_btn.onRelease = function() {
  fscommand("popup", "http://www.adobe.com/");
};
alert_btn.onRelease = function() {
  fscommand("alert", "You clicked the button.");
};
```

5. Sélectionnez Fichier > Paramètres de publication et assurez-vous que Flash avec FSCommand est sélectionné dans le menu Modéle sur l'onglet HTML.
6. Sélectionnez Fichier > Publier pour générer les fichiers SWF et HTML.
7. Dans un éditeur de texte ou HTML, ouvrez le fichier HTML qui a été généré à l'étape 6, puis examinez le code. Lorsque vous avez publié votre fichier SWF à l'aide du modèle Flash avec FSCommand dans l'onglet HTML de la boîte de dialogue Paramètres de publication, du code supplémentaire a été inséré dans le fichier HTML. Les attributs `NAME` et `ID` du fichier SWF forment le nom de fichier. Par exemple, pour le fichier myMovie.fla, les attributs seront définis sur `myMovie`.
8. Dans le fichier HTML, ajoutez le code JavaScript suivant lorsque le document indique

```
// Placez votre code ici. :

if (command == "alert") {
  alert(args);
} else if (command == "popup") {
  window.open(args, "mmwin", "width=500,height=300");
}
```

(Pour plus d'informations au sujet de de la publication, consultez le guide *Utilisation de Flash*.)

Sinon, pour les applications Microsoft Internet Explorer, vous pouvez associer un gestionnaire d'événement directement dans la balise `<SCRIPT>` comme l'illustre cet exemple :

```
<script Language="JavaScript" event="FSCommand (command, args)"
  for="theMovie">
...
</script>
```

9. Sauvegardez et fermez le fichier HTML.

   Quand vous modifiez les fichiers HTML en dehors de Flash de cette façon, rappelez-vous que vous devez déselectionner la case à cocher HTML dans Fichier > Paramètres de publication, faute de quoi votre code HTML sera écrasé par Flash lors d'une nouvelle publication.

10. Dans un navigateur Web, ouvrez le fichier HTML à afficher. Cliquez le bouton Ouvrir fenêtre ; le site Web de Macromedia s'affiche dans une fenêtre distincte. Cliquez sur le bouton Alerte ; une fenêtre d'alerte apparaît.

La fonction `fscommand()` peut envoyer des messages à Macromedia Director qui sont interprétés par Lingo comme des chaînes, des événements ou un code Lingo exécutable. Si le message est une chaîne ou un événement, vous devez écrire le code Lingo pour le recevoir depuis la fonction `fscommand()` et entraîner une action dans Director. Pour plus d'informations, consultez le centre de support de Director à l'adresse www.adobe.com/support/director.

En Visual Basic, Visual C++ et d'autres programmes pouvant héberger les contrôles ActiveX, `fscommand()` envoie un événement VB avec deux chaînes qui peut être traité dans le langage de programmation de l'environnement. Pour plus d'informations, utilisez les mots-clés *méthode Flash* pour effectuer une recherche dans le centre d'assistance de Flash à l'adresse http://www.adobe.com/fr/support/flash/.

Utilisation de JavaScript pour contrôler les applications Flash

Flash Player 6 (6.0.40.0) et les versions ultérieures prennent en charge certaines méthodes JavaScript qui sont spécifiques aux applications Flash, ainsi que `FSCommand` dans Netscape 6.2 et ultérieure. Les versions antérieures ne prennent pas en charge les méthodes JavaScript et `FSCommand` dans Netscape 6.2 ou versions ultérieures. Pour plus d'informations, consultez l'article du Centre de support d'Adobe, « Scripting With Flash », sur le site www.adobe.com/support/flash/publishexport/scriptingwithflash/.

Pour Netscape 6.2 et versions ultérieures, vous n'avez pas besoin de définir l'attribut `swliveconnect` sur la valeur `true`. Cependant, la définition de `swLiveConnect` sur la valeur `true` n'a aucune incidence sur votre fichier SWF. Pour plus d'informations, consultez la description de l'attribut `swLiveConnect` dans le guide *Utilisation de Flash*.

Présentation des méthodes Flash Player

Vous pouvez utiliser les méthodes Flash Player pour contrôler un fichier SWF dans Flash Player avec des langages tels que JavaScript et VBScript. Comme avec les autres méthodes, vous pouvez utiliser les méthodes Flash Player pour envoyer des appels à des fichiers SWF depuis un environnement de programmation autre qu'ActionScript. Chaque méthode possède un nom et la plupart prennent des paramètres. Un paramètre spécifie une valeur sur laquelle opère la méthode. Le calcul effectué par certaines méthodes renvoie une valeur qui peut être utilisée par l'environnement de programmation.

Deux technologies permettent la communication entre le navigateur et Flash Player : LiveConnect (Netscape Navigator 3.0 et versions ultérieures sous Windows 95/98/2000/NT/XP ou Power Macintosh) et ActiveX (Internet Explorer 3.0 et versions ultérieures sous Windows 95/98/2000/NT/XP). Bien que les techniques de programmation soient équivalentes pour tous les navigateurs et les langages, des propriétés et événements supplémentaires sont disponibles pour l'utilisation des contrôles ActiveX.

Pour obtenir plus d'informations, ainsi que la liste complète des méthodes de programmation de Flash Player, utilisez les mots-clés *méthode Flash* pour rechercher le centre d'assistance de Flash à l'adresse www.adobe.com/fr/support/flash.

A propos de l'API externe

La classe ExternalInterface est aussi appelée *API Externe*, qui est un nouveau sous-système qui vous laisse facilement communiquer depuis ActionScript et le conteneur Flash Player à une page HTML avec JavaScript ou à une application de bureau qui incorpore Flash Player.

REMARQUE

Cette fonctionalité remplace l'ancienne fonction `fscommand()` pour interopérer avec une page HTML ou une application conteneur. L'API externe offre une fonctionalité plus robuste que `fscommand()` dans cette situation. Pour plus d'informations, voir « A propos de l'API externe », à la page 665.

La classe ExternalInterface n'est que disponible dans les circonstances suivantes :

- Dans toutes les versions prises en charge d'Internet Explorer pour Windows (5.0 et ultérieure).
- Dans un conteneur ActiveX personnalisé et incorporé, tel qu'une application de bureau incorporant le contrôle ActiveX de Flash Player.
- Dans tout navigateur qui prend en charge l'interface NPRuntime (qui actuellement comprend les navigateurs suivants :
  - Firefox 1.0 et ultérieure
  - Mozilla 1.7.5 et ultérieure
  - Netscape 8.0 et ultérieure
  - Safari 1.3 et ultérieure.

Dans toutes les autres situations, la propriété `ExternalInterface.available` renvoie une erreur.

Depuis ActionScript, vous pouvez appeler une fonction JavaScript sur la page HTML. L'API externe apporte les améliorations fonctionnelles suivantes grâce à `fscommand()` :

- Vous pouvez utiliser toute fonction JavaScript, pas seulement les fonctions que vous pouvez utiliser avec la `fonction fscommand`.
- Vous pouvez transmettre n'importe quel nombre d'arguments, avec n'importe quels noms ; vous n'êtes pas limité à transmettre une commande et des arguments.
- Vous pouvez transmettre des types de données variées (telles que Booléenne, Nombre, et Chaîne); vous n'êtes plus limité aux paramètres de Chaîne.
- Vous pouvez maintenant recevoir la valeur d'un appel et cette valeur retourne immédiatement à ActionScript (comme la valeur de retour d'un appel que vous faites).

Vous pouvez appeler une fonction ActionScript depuis JavaScript sur une page HTML Pour plus d'informations, consultez `ExternalInterface (flash.external.ExternalInterface)`. Pour plus d'informations sur la sécurité de fichiers locaux, consultez « Sécurité des fichiers locaux et Flash Player », à la page 677.

Les sections suivantes contiennent des exemples qui utilisent l'API externe :

- « Création d'une interaction avec l'API externe », à la page 667
- « Contrôle de Flash Video avec l'API externe », à la page 670

Création d'une interaction avec l'API externe

Vous pouvez créer l' interaction entre le navigateur et un fichier SWF qui est incorporé dans une page Web. La procédure suivante envoie le texte à la page HTML qui contient le fichier SWF et le HTML envoie un message en retour au fichier SWF lors de l'exécution.

Pour créer l'application Flash :

1. Créez un document Flash, puis enregistrez-le sous le nom de **extint.fla**.
2. Faites glisser deux composants TextInput sur la Scène et nommez leurs occurrences **in_ti** et **out_ti**.
3. Faites glisser un composant Label sur la scène, nommez son occurrence **out_lbl**, positionnez-le au-dessus de l'occurrence TextInput `out_ti`, et paramétrez la propriété du texte dans l'onglet Paramètres de l'inspecteur Propriétés sur **Sending to JS:**.
4. Faites glisser un composant Button sur la Scène, positionnez le à côté du label `out_lbl` et nommez son occurrence **send_button**.
5. Faites glisser un composant Label sur la Scène, nommez son occurrence **out_lbl**, positionnez-le au-dessus de l'occurrence TextInput `out_ti`, et paramétrez la propriété du texte dans l'onglet Paramètres sur **Receiving from JS:**.
6. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
import flash.external.ExternalInterface;

ExternalInterface.addCallback("asFunc", this, asFunc);
function asFunc(str:String):Void {
  in_ti.text = "JS > Hello " + str;
}

send_button.addEventListener("click", clickListener);
function clickListener(eventObj:Object):Void {
  trace("click > " + out_ti.text);
  ExternalInterface.call("jsFunc", out_ti.text);
}
```

Ce code se compose de trois sections différentes. La première section importe la classe ExternalInterface ainsi vous n'avez pas à utiliser son nom de classe entièrement qualifié. La deuxième section du code définit une fonction de rappel, `asFunc()`, qui est nommée à partir de JavaScript dans un document HTML créé dans un exemple qui va suivre. Cette fonction paramètre le texte dans un composant TextInput sur la scène. La troisième section de code définit une fonction et l'affecte en tant qu'événement écouteur pour quand l'utilisateur clique sur l'occurrence du composant Button sur la scène. Lorsque l'utilisateur clique sur le bouton, le fichier SWF appelle la fonction JavaScript `jsFunc()` dans la page HTML et transmet la propriété du texte de l'occurrence `out_ti` d'entrée du texte.

7. Sélectionnez Fichier > Paramètres de publication, puis sélectionnez l'onglet Formats et assurez-vous que Flash et HTML sont sélectionnés.
8. Cliquez sur Publier pour créer les fichiers HTML et SWF.

   Quand c'est fait, continuez à la procédure suivante afin de créer le conteneur pour le fichier SWF.

Avant de tester le document Flash précédent, vous devez modifier le code HTML généré et ajouter du code HTML et JavaScript. La procédure modifie le conteneur HTML pour le fichier SWF de sorte que les deux fichiers puissent interagir quand ils sont exécutés dans un navigateur.

Pour créer le conteneur HTML pour le fichier SWF :

1. Complétez la procédure précédente.
2. Ouvrez le fichier extint.html que Flash crée quand vous publiez l'application.

   Il est dans le même dossier que le document Flash.
3. Ajoutez le code JavaScript suivant entre les balises d'ouverture et de fermeture :

```
<script language="JavaScript">
<!--
  function thisMovie(movieName) {
    var isIE = navigator.appName.indexOf("Microsoft") != -1;
    renvoie (isIE) ? window[movieName] : document[movieName];
  }

  function makeCall(str) {
    thisMovie("extint").asFunc(str);
  }

  function jsFunc(str) {
    document.inForm.inField.value = "AS > Hello " + str;
  }
// -->
</script>
```

   Ce code JavaScript définit trois méthodes. La première méthode renvoie une référence au fichier encastré SWF selon si le navigateur de l'utilisateur est Microsoft Internet Explorer (IE) ou Mozilla. La deuxième fonction, `makeCall()`, appelle la méthode `asFunc()` que vous avez définie dans le document Flash dans l'exemple précédent. Le paramètre `"extint"` de l'appel de fonction `thisMovie()` fait référence à l'ID objet et incorpore le nom du fichier SWF incorporé. Si vous avez sauvegardé le document Flash avec un nom différent, vous avez besoin de modifier cette chaîne pour faire correspondre les valeurs dans les balises object et embed. La troisième fonction, `jsFunc()`, établit la valeur du champ texte `inField` dans le document HTML. Cette fonction est appelée depuis le document Flash quand un utilisateur clique sur le bouton `send_button`.

4. Ajoutez le code HTML suivant avant la balise de fermeture, `</body>` :

```
<form name="outForm" method="POST"
  action="javascript:makeCall(document.outForm.outField.value);">
  Sending to AS:<br />
  <input type="text" name="outField" value="" /><br />
  <input type="submit" value="Send" />
</form>

<form name="inForm" method="POST" action="">
  Receiving from AS:<br />
  <input type="text" name="inField">
</form>
```

   Ce code HTML crée deux formulaires HTML analogues aux formulaires créés dans l'environnement Flash lors de l'exercice précédent. Le premier formulaire soumet la valeur du champ `outField` de texte à la fonction JavaScript `makeCall()` défini lors d'une étape précédente. Le deuxième formulaire est utilisé pour afficher une valeur qui est envoyée depuis le fichier SWF quand l'utilisateur clique sur l'occurrence `send_button`.

5. Sauvegardez le document HTML et téléchargez les fichiers HTML et SWF vers le serveur web.

6. Visualisez le fichier HTML dans un navigateur Web, entrez une chaîne dans l'occurence `out_ti` TextInput et cliquez le bouton Envoyer.

   Flash appelle la fonction `jsFunc()` JavaScript et transmet les contenus du champ de texte `out_ti`, qui affiche les contenus dans le champ de texte d'entrée du formulaire HTML `inForm` `inField`.

7. Tapez une valeur dans le champ de texte HTML `outField` et cliquez le bouton Envoyer.

   Flash appelle la fonction du fichier SWF `asFunc()` qui affiche la chaîne dans l'occurrence `in_ti` TextInput.

Pour un exemple de fichier source, ExtInt.fla, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/ExternalAPI afin d'accéder à cet exemple.

Pour un exemple plus complexe qui utilise l'API externe, consultez « Contrôle de Flash Video avec l'API externe », à la page 670. Pour plus d'informations sur la sécurité de fichiers locaux, consultez « Sécurité des fichiers locaux et Flash Player », à la page 677.

REMARQUE

Evitez d'utiliser d'autres méthodes pour accéder à l'objet plug-in, tel que `document.getElementById("pluginName")` or `document.all.pluginName` parce que ces autres méthodes ne fonctionnent pas de manière homogène sur tous les navigateurs.

Contrôle de Flash Video avec l'API externe

La procédure suivante vous montre comment contrôler les fichiers Flash Video (FLV) en utilisant des contrôles dans une page HTML et affiche des informations sur le fichier vidéo dans un champ de texte HTML. Cette procédure utilise l'API externe pour obtenir cette fonctionnalité.

Pour construire une application Flash avec l'API externe :

1. Créez un document Flash, puis enregistrez-le sous le nom de **video.fla**.
2. Ajoutez un nouveau symbole vidéo à la bibliothèque en sélectionnant Nouvelle vidéo dans le menu contextuel du panneau Bibliothèque.
3. Faites glisser le symbole vidéo sur la scène et donnez-lui un nom d'occurrence de **selected_video**.
4. Sélectionnez l'occurrence `selected_video` et puis l'inspecteur de propriété pour redimensionner l'occurrence à **320** pixels de largeur par **240** pixels de hauteur.
5. Définissez les coordonnées *x* et *y* pour la position de la vidéo sur **0**.
6. Sélectionnez la scène et utilisez l'inspecteur Propriétés pour redimensionner à **320** pixels par **240** pixels.

   Maintenant la scène correspond aux dimensions de l'occurrence vidéo.
7. Ajoutez le code ActionScript suivant à l'image 1 du scénario principal :

```
import flash.external.ExternalInterface;

/* Enregistrer playVideo() et pauseResume() afin que ceci soit possible
pour les appeler à partir de JavaScript dans la page HTML de conteneur.
  */
ExternalInterface.addCallback("playVideo", null, playVideo);
ExternalInterface.addCallback("pauseResume", null, pauseResume);

/* La vidéo nécessite un objet NetConnection et NetStream. */
var server_nc:NetConnection = new NetConnection();
server_nc.connect(null);
var video_ns:NetStream = new NetStream(server_nc);

// Associer l'objet NetStream à l'objet Video sur la scène afin
que les données NetStream soient affichées dans l'objet Video. */
selected_video.attachVideo(video_ns);
```

```
/* La méthode onStatus() est appelée automatiquement lorsque le statut de
  l'objet NetStream est mis à jour (la vidéo commence la lecture, par
  exemple). Lorsque ceci se produit, envoyez la valeur de la propriété
  de code à la page HTML en appelant la fonction updateStatus() de
  JavaScript via ExternalInterface. */
video_ns.onStatus = function(obj:Object):Void {
  ExternalInterface.call("updateStatus", "  " + obj.code);
};

function playVideo(url:String):Void {
  video_ns.play(url);
}

function pauseResume():Void {
  video_ns.pause();
}
```

La première partie de ce code ActionScript définit deux fonctions de rappel ExternalInterface, `playVideo()` et `pauseResume()`. Ces fonctions sont appelées à partir de JavaScript dans la procédure suivante. La seconde partie du code crée un nouvel objet NetConnection et NetStream, que vous utilisez avec l'occurrence vidéo pour lire dynamiquement les fichiers FLV.

Le code dans la procédure suivante définit un gestionnaire d'événement `onStatus` pour l'objet NetStream `video_ns`. A chaque fois que l'objet NetStream change son statut, Flash utilise la méthode `ExternalInterface.call()` pour déclencher la fonction JavaScript personnalisée, `updateStatus()`. Les deux dernières fonctions, `playVideo()` et `pauseResume()`, contrôlent la lecture de l'occurrence vidéo sur la scène. Ces deux fonctions sont appelées à partir de JavaScript écrit la procédure suivante.

8. Enregistrez le document Flash.
9. Sélectionner Fichier > Paramètres de publication, puis sélectionnez l'onglet Formats et assurez-vous que Flash et HTML sont tous les deux sélectionnés.
10. Cliquez sur Publier pour publier les fichiers HTML et SWF sur votre disque dur.

    Quand c'est fait, continuez à la procédure suivante afin de créer le conteneur pour le fichier SWF.

Pour un exemple de fichier source, external.fla, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/ExternalAPI afin d'accéder à l'exemple.

Dans la procédure suivante, vous modifiez le code HTML généré par Flash dans la procédure précédente. Cette procédure crée le JavaScript et HTML requis que les fichiers FLV soient lus dans le fichier SWF.

Pour créer le conteneur HTML pour le fichier SWF :

1. Complétez la procédure précédente.
2. Ouvrez le document video.html que vous avez publié dans la dernière étape de la procédure précédente.
3. Modifiez le code existant afin qu'il corresponde au code suivant.

REMARQUE	Examinez les commentaires du code dans l'exemple suivant. Un exposé sur le code suit l'exemple de code.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
  www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"
  />
<title>ExternalInterface</title>

<script language="JavaScript">
  // Utilisez une variable pour référencer le fichier incorporé d
  // le fichier SWF.
  var flashVideoPlayer;

  /* Lorsque la page HTML se charge (à travers l'événement onLoad du
  marqueur <body>), il appelle la fonction initialize(). */
  function initialize() {
    /* Vérifiez si le navigateur est IE. S'il y a lieu, flashVideoPlayer
       est window.videoPlayer. Autrement, c'est document.videoPlayer. Le
       videoPlayer est l'ID attribué aux marqueurs <object> et <embed>.
       */
    var isIE = navigator.appName.indexOf("Microsoft") != -1;
    flashVideoPlayer = (isIE) ? window['videoPlayer'] :
  document['videoPlayer'];
  }

  /* Lorsque l'utilisateur clique sur le bouton lire dans le formulaire,
  met à jour la zone de texte videoStatus, et appelle la fonction
  playVideo() dans le fichier SWF, lui passant l'URL du fichier FLV. */
  function callFlashPlayVideo() {
    var comboBox = document.forms['videoForm'].videos;
    var video = comboBox.options[comboBox.selectedIndex].value;
    updateStatus("____" + video + "____");
    flashVideoPlayer.playVideo("http: www.helpexamples.com/flash/video/
  " + video);
  }
```

```
    // Appelle la fonction pauseResume() dans le fichier SWF.
    function callFlashPlayPauseVideo() {
      flashVideoPlayer.pauseResume();
    }

    /* La fonction updateStatus() est appelée à partir du fichier SWF de
    la méthode onStatus() de l'objet NetStream. */
    function updateStatus(message) {
      document.forms['videoForm'].videoStatus.value += message + "\n";
    }
  </script>
  </head>
  <body bgcolor="#ffffff" onLoad="initialize();">

  <object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
    codebase="http://fpdownload.adobe.com/pub/shockwave/cabs/flash/
    swflash.cab#version=8,0,0,0" width="320" height="240" id="videoPlayer"
    align="middle">
  <param name="allowScriptAccess" value="sameDomain" />
  <param name="movie" value="video.swf" />
  <param name="quality" value="high" />
  <param name="bgcolor" value="#ffffff" />
  <embed src="video.swf" quality="high" bgcolor="#ffffff" width="320"
    height="240" name="videoPlayer" align="middle"
    allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
    pluginspage="http://www.adobe.com/go/getflashplayer" />
  </object>

  <form name="videoForm">
    Select a video:<br />
    <select name="videos">
      <option value="lights_long.flv">lights_long.flv</option>
      <option value="clouds.flv">clouds.flv</option>
      <option value="typing_long.flv">typing_long.flv</option>
      <option value="water.flv">water.flv</option>
    </select>
    <input type="button" name="selectVideo" value="play"
    onClick="callFlashPlayVideo();" />
    <br /><br />

    Playback <input type="button" name="playPause" value="play/pause"
    onClick="callFlashPlayPauseVideo();" />

    <br /><br />
    Video status messages <br />
    <textarea name="videoStatus" cols="50" rows="10"></textarea>
  </form>

  </body>
  </html>
```

Ce code HTML définit quatre fonctions JavaScript : `initialize()`, `callFlashPlayVideo()`, `callFlashPlayPauseVideo()`, et `updateStatus()`.
La fonction `initialize()` est appelée dans le marqueur `body` dans l'événement `onLoad`.
Les fonctions `callFlashPlayVideo()` et `callFlashPlayPauseVideo()` sont toutes deux appelées lorsque l'utilisateur clique sur ou bien le bouton lire ou lire/pause dans le document HTML, et déclenche les fonctions `playVideo()` et `pauseResume()` dans le fichier SWF.

La fonction finale, `updateStatus()`, est appelée par le fichier SWF à chaque fois que le gestionnaire d'événement `onStatus` de l'objet NetStream `video_ns` est déclenché.
Ce code HTML définit aussi un formulaire qui a une liste déroulante de vidéos à partir de laquelle l'utilisateur peut choisir. A chaque fois que l'utilisateur sélectionne une vidéo et clique sur le bouton lire, la fonction JavaScript `callFlashPlayVideo()` est appelée, qui appelle ensuite la fonction `playVideo()` dans le fichier SWF. La fonction passe l'URL du fichier SWF à charger dans l'occurrence vidéo. Tandis que la vidéo est lue et que le statut de l'objet NetStream change, le contenu de la zone de texte HTML sur la scène est mis à jour.

4. Sauvegardez vos modifications au document HTML et puis chargez les fichiers HTML et SWF vers le serveur web .
5. Ouvrez le document video.html à distance à partir du site Web, sélectionnez une vidéo de la liste déroulante et cliquez sur le bouton lire.

   Flash lit le fichier FLV sélectionné et met à jour le contenu de la zone de texte videoStatus dans le document HTML.

Pour un exemple de fichier source, external.fla, consultez la page des exemples Flash à l'adresse www.adobe.com/go/learn_fl_samples_fr. Téléchargez et décompressez le fichier zip Exemples et naviguez jusqu'au dossier ActionScript2.0/ExternalAPI afin d'accéder à l'exemple.

Pour plus d'informations sur l'API externe, consultez la section `ExternalInterface (flash.external.ExternalInterface)` dans le *Guide de référence du langage ActionScript 2.0*.

Pour plus d'informations sur la sécurité de fichiers locaux, consultez « Sécurité des fichiers locaux et Flash Player », à la page 677.

REMARQUE	Evitez d'utiliser d'autres méthodes pour accéder à l'objet plug-in, tel que document.getElementById("pluginName") ou document.all.pluginName, parce que ces autres méthodes ne fonctionnent pas de manière homogène sur tous les navigateurs.

CHAPITRE 16

Fonctionnement de la sécurité

16

Dans Flash CS3 Professional, vous pouvez utiliser ActionScript pour charger des données provenant de sources externes dans un fichier SWF ou envoyer des données vers un serveur. Lorsque vous chargez les données dans un fichier SWF, vous devez comprendre le modèle de sécurité Flash et en tenir compte. Lorsque vous ouvrez un fichier SWF sur votre disque dur, vous pourriez avoir besoin de paramétrer des configurations particulières afin de pouvoir tester votre fichier localement.

Pour plus d'informations sur la sécurité des fichiers locaux, consultez « Sécurité des fichiers locaux et Flash Player », à la page 677. Pour plus d'informations sur les changements apportés au modèle de sécurité lors la création de la version 8 et versions ultérieures, consultez la section « Compatibilité avec les modèles précédents de sécurité Flash Player », à la page 676. Pour plus d'informations sur le chargement et l'analyse de données à partir d'un serveur, consultez le Chapitre 15, « Utilisation de données externes », à la page 631. Pour plus d'informations sur la sécurité, consultez le site www.adobe.com/devnet/security et www.adobe.com/software/flashplayer/security/.

Pour plus d'informations sur la sécurité dans Flash 8 et versions ultérieures, consultez les sections suivantes :

Compatibilité avec les modèles précédents de sécurité Flash Player

Suite aux changements des fonctions de sécurité dans Flash Player 7, un contenu qui fonctionne comme prévu sous Flash Player 6 (ou une version antérieure) peut ne pas s'exécuter comme prévu sous des versions de Flash Player ultérieures. Par exemple, dans Flash Player 6, un fichier SWF se trouvant dans www.adobe.com peut lire des données sur un serveur situé à l'adresse data.adobe.com. Ainsi, Flash Player 6 autorisait un fichier SWF provenant d'un domaine à charger des données à partir d'un domaine similaire.

Dans Flash Player 7 et les versions ultérieures, si un fichier SWF version 6 (ou antérieure) tente de charger des données à partir d'un serveur se trouvant dans un autre domaine et ne fournissant aucun fichier de régulation autorisant la lecture à partir du domaine de ce fichier SWF, la boîte de dialogue Paramètres de Flash Player apparaît. L'utilisateur doit alors autoriser ou refuser l'accès aux données interdomaines.

Si l'utilisateur clique sur Autoriser, le fichier SWF peut accéder aux données requises ; s'il clique sur Refuser, le fichier SWF ne peut pas accéder aux données requises.

Pour empêcher l'apparition de cette boîte de dialogue, vous devez créer un fichier de régulation de sécurité sur le serveur fournissant les données. Pour plus d'informations, voir « Autorisation de chargement de données inter-domaines », à la page 706.

Flash Player 7 et les versions ultérieures ne permettent pas l'accès inter-domaines sans fichier de régulation de sécurité.

Flash Player 8 et les versions ultérieures ont changé la façon dont il manipule `System.security.allowDomain`. Un fichier Flash SWF (Flash Player 8 et versions ultérieures) qui appelle `System.security.allowDomain` avec n'importe quel argument ou tout autre type de fichier SWF qui utilise le caractère générique (*), permet uniquement l'accès à lui-même. Il y a maintenant la prise en charge pour une valeur de caractère générique (*) par exemple : `System.security.allowDomain("*")` et `System.security.allowInsecureDomain("*")`. Si un fichier SWF de la version 7 ou antérieure appelle `System.security.allowDomain` ou `System.security.allowInsecureDomain` avec un argument autre que le caractère générique (*), ceci affectera tous les fichiers SWF de la version 7 ou antérieure dans le domaine du fichier SWF d'appel, comme dans Flash Player 7. Cependant, cette sorte d'appel n'affecte aucun fichier SWF Flash Player 8 (ou une version ultérieure) dans le domaine du fichier SWF d'appel. Ceci aide à réduire l'interruption du contenu patrimonial dans Flash Player.

Pour plus d'informations, consultez la section « Restriction des API de réseau », à la page 695 et les entrées allowDomain (méthode security.allowDomain) et allowInsecureDomain (méthode security.allowInsecureDomain).

Flash Player 8 et versions ultérieures ne permettent pas aux fichiers SWF locaux de communiquer avec Internet sans une configuration spécifique sur votre ordinateur. Supposez que vous disposez de contenu patrimonial publié avant la mise en application de ces restrictions. Si ce contenu tente de communiquer avec le réseau ou le système de fichiers local, Flash Player interrompt l'opération, et vous devez fournir une autorisation de façon explicite pour que l'application fonctionne correctement. Pour plus d'informations, voir « Sécurité des fichiers locaux et Flash Player », à la page 677.

Sécurité des fichiers locaux et Flash Player

Dans le modèle de sécurité de Flash Player, les applications Flash et les fichiers SWF sur un ordinateur local ne sont autorisés à communiquer *ni* avec Internet ni avec le système de fichiers local par défaut. Un *fichier SWF local* est un fichier SWF installé localement sur l'ordinateur d'un utilisateur, pas obtenu via un site Web et qui ne comprend pas de fichiers de projections (EXE).

REMARQUE	Les restrictions qui sont discutées dans cette section n'affectent pas les fichiers SWF qui sont sur Internet.

Lorsque vous créez un fichier FLA, vous pouvez indiquer s'il est autorisé à communiquer avec un réseau ou un système de fichiers local. Dans les versions précédentes de Flash Player (7 et antérieures), les fichiers SWF locaux peuvent interagir avec d'autres fichiers SWF et charger des données à partir de n'importe quel emplacement à distance ou local. Dans Flash Player 8 et versions ultérieures, un fichier SWF ne peut pas effectuer de connexion avec le système de fichiers local *et* avec Internet. Ceci est un changement de sécurité, afin qu'un fichier SWF ne puisse pas lire de fichier sur votre disque dur et puis envoyer le contenu de ces fichiers à travers Internet.

Cette restriction de sécurité affecte un contenu déployé localement, qu'il soit patrimonial (un fichier FLA créé dans une version antérieure de Flash) ou qu'il ait été créé dans Flash 8 et ses versions ultérieures. Supposez que vous déployez une application Flash, en utilisant Flash MX 2004 ou antérieur, qui s'exécute localement et accède aussi à Internet. Dans Flash Player 8 et les versions ultérieures, cette application demande désormais à l'utilisateur l'autorisation de communiquer avec Internet.

Lorsque vous testez un fichier sur votre disque dur, il y a une série d'étapes afin de déterminer si le fichier est un document local approuvé ou un document potentiellement non approuvé. Si vous créez le fichier dans l'environnement de programmation de Flash par exemple, lorsque vous sélectionnez Contrôle > Tester l'animation, votre fichier est de confiance parce que c'est un environnement de test.

Dans Flash Player 7 et version antérieure, les fichiers SWF locaux étaient autorisés à lire à partir du système de fichiers local et du réseau (tel qu'Internet). Dans Flash Player 8 et les versions ultérieures, les fichiers SWF locaux peuvent disposer des niveaux d'autorisation suivants :

Accès au système de fichiers local uniquement (par défaut) Un fichier SWF local peut lire le système de fichiers local et les chemins réseau de règle d'affectation des noms (UNC) mais pas communiquer avec Internet. Pour plus d'informations sur les fchiers SWF avec accès aux fichierx locaux, consultez la section « Accès aux fichiers locaux uniquement (par défaut) », à la page 686.

Accéder au réseau uniquement Un fichier SWF local peut accéder au réseau (tel qu'Internet) mais pas au système de fichiers local sur lequel il est installé. Pour plus d'informations sur les fichiers SWF avec accès au réseau uniquement, consultez la section « Accès au réseau uniquement », à la page 687.

Accès au système de fichiers local et au réseau Un fichier SWF local peut lire à partir du système de fichiers local sur lequel il est installé, lire et écrire sur et à partir de serveurs, et inter-coder d'autres fichiers SWF du réseau ou du système de fichiers local. Ces fichiers sont considérés comme de confiance, et se comportent comme dans Flash Player 7. Pour plus d'informations sur les fichiers SWF d'accès local et de réseau, consultez la section « Accès au système de fichiers et au réseau », à la page 687.

Pour plus d'informations sur la sécurité de fichiers locaux dans Flash 8 et versions ultérieures se rapportant à l'outil de programmation, consultez les sections suivantes :

- « Fonctionnement des Sandbox de sécurité locale », à la page 679
- « Paramètres de sécurité de Flash Player », à la page 680
- « Sécurité des fichiers locaux et fichiers de projections », à la page 682
- « Dépannage des fichiers SWF patrimoniaux », à la page 683
- « Correction du contenu patrimonial déployé sur des ordinateurs locaux », à la page 684
- « Publication des fichiers pour le déploiement local », à la page 685

Pour plus d'informations sur la sécurité de fichiers locaux pour les utilisateurs, consultez « Paramètres de sécurité de Flash Player », à la page 680. Pour plus d'informations sur la sécurité, consultez le site www.adobe.com/devnet/security/ et www.adobe.com/software/flashplayer/security/.

Fonctionnement des Sandbox de sécurité locale

Il existe plusieurs Sandbox de sécurité différents Flash Player. Chacun détermine comment un fichier SWF peut interagir avec le système de fichiers local, le réseau, ou les deux en même temps. Ils limitent la façon dont un fichier peut interagir avec le système de fichiers local, le réseau, ou les deux en même temps. Comprendre les Sandbox de sécurité vous aide à developper et tester les applications Flash sur votre ordinateur sans rencontrer d'erreurs inattendues.

Local-with-file-system

Pour des raisons de sécurité, Flash Player place tous les fichiers SWF locaux, y compris les fichiers SWF locaux patrimoniaux, dans le Sandbox local-with-file-system, par défaut (sauf si une autre configuration est effectuée). Pour certains fichiers SWF patrimoniaux (Flash Player 7 et versions antérieures), les opérations peuvent être affectées par les restrictions applicables sur leurs accès (pas en dehors de l'accès réseau), mais ceci fourni le paramètre par défaut le plus sécurisé pour la protection de l'utilisateur.

A partir de ce Sandbox, les fichiers SWF peuvent lire les fichiers sur le système de fichiers local ou les chemins réseau UNC (en utilisant la méthode `XML.load()`, par exemple), mais ils ne peuvent en aucune manière communiquer avec le réseau. Ceci garanti à l'utilisateur que les données locales ne peuvent pas filtrer hors du réseau ou autrement être partagées de manière inopportune.

Local-with-networking

Lorsque les fichiers SWF locaux sont attribués au Sandbox local-with-networking, ils perdent leur accès au système de fichiers local. En retour, les fichiers SWF sont autorisés à accéder au réseau. Cependant, un fichier SWF local-with-networking n'est toujours pas autorisé à lire n'importe quelle donnée dérivée du réseau sauf si des permissions sont présentes pour cette action. Par conséquent, un fichier SWF local-with-networking n'a aucun accès local, mais il peut transmettre des données sur le réseau et lire les données réseau à partir des sites pour lesquels il a des permission d'accès spécifiques.

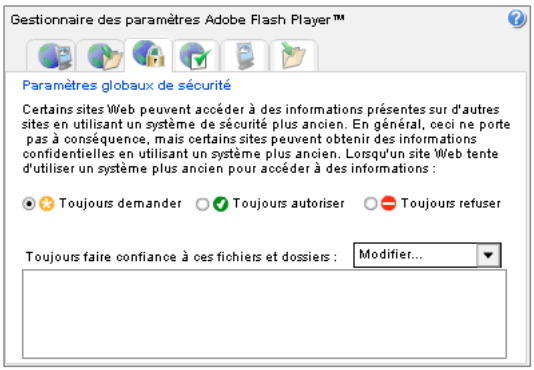
Local-trusted

Les fichiers SWF attribués au Sandbox local-trusted peuvent interagir avec tous les autres fichiers SWF et charger des données à partir de n'importe quel emplacement (à distance ou localement).

Paramètres de sécurité de Flash Player

Adobe a conçu Flash Player de façon à réduire les autorisations d'accès manuelles au strict nécessaire. Il demeure un certain nombre de contenus Flash patrimoniaux qui ont été créés avec des règles de sécurité pour Flash Player 7 ou antérieur. Dans ces cas, Flash Player vous permet d'autoriser ce contenu à s'exécuter conformément aux règles d'origine, ou vous pouvez appliquer les règles les plus récentes, qui sont plus strictes. Cette dernière possibilité permet de s'assurer que tous les contenus que vous consultez répondent aux normes de sécurité les plus récentes, ce qui risque d'entraver la lecture des contenus plus anciens.

Tous les utilisateurs qui affichent les fichiers SWF (y compris les développeurs non Flash) peuvent définir des autorisations globales via le panneau Paramètres globaux de sécurité du gestionnaire de paramètres de Flash Player (illustré dans la figure suivante).

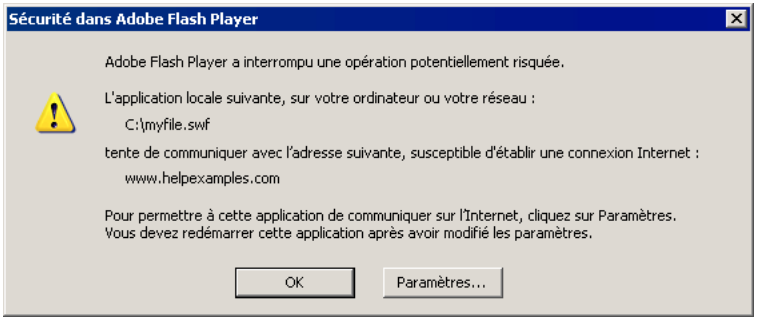


Lorsque du contenu ancien est lu par une version plus récente du programme, vous devez déterminer les règles qui s'appliquent. L'une des boîtes de dialogue suivantes peut alors s'afficher. Ces boîtes de dialogue vous permettent d'autoriser manuellement la communication avec d'autres emplacements Internet des contenus Flash plus anciens :

- Sous certaines conditions, une boîte de dialogue s'affiche pour indiquer que le contenu Flash que vous utilisez tente d'utiliser les anciennes règles de sécurité à partir d'un site qui n'appartient pas au même domaine, ce qui risque d'entraîner le partage des informations entre plusieurs sites. Vous devez alors spécifier si vous souhaitez autoriser ou refuser ce type d'accès.

  En supplément de la boîte de dialogue de confirmation, utilisez le panneau Paramètres globaux de sécurité pour spécifier si Flash Player doit toujours vous demander une autorisation d'accès, par l'intermédiaire d'une boîte de dialogue, toujours refuser l'accès, sans confirmation, ou toujours autoriser l'accès aux autres sites ou domaines, sans confirmation.

- (Flash Player 8 et versions ultérieures) Une boîte de dialogue peut s'afficher pour vous indiquer qu'un fichier SWF tente de communiquer sur Internet. Flash Player ne permet pas au contenu Flash local de communiquer avec Internet, par défaut.



  Cliquez sur Paramètres pour accéder au panneau Paramètres globaux de sécurité, qui permet de spécifier les applications Flash, enregistrées sur votre ordinateur, qui peuvent communiquer sur Internet.

Pour modifier les paramètres de sécurité ou plus de détails sur vos options, vous utilisez le panneau Paramètres globaux de sécurité. Utilisez ce panneau pour réinitialiser les paramètres de contrôle de l'accès dans Adobe Flash Player :

- Si vous sélectionnez *Toujours refuser* et que vous confirmez votre choix, tout site Web tentant d'utiliser votre caméra ou microphone se verra refuser l'accès. Vous ne recevez plus de demande d'autorisation de l'accès à votre caméra ou microphone. Cette action s'applique à la fois aux sites Web que vous avez déjà consultés et à ceux que vous n'avez jamais consultés.
- Si vous sélectionnez *Toujours demander* et que vous confirmez votre choix, tout site Web tentant d'utiliser votre caméra ou microphone doit vous demander l'autorisation. Cette action s'applique à la fois aux sites Web que vous avez déjà consultés et à ceux que vous n'avez jamais consultés.

Si vous avez préalablement sélectionné Mémoriser dans le panneau Paramètres de contrôle de l'accès (voir la figure suivante) pour autoriser ou refuser de manière permanente l'accès à un ou plusieurs sites Web, sélectionnez Toujours demander ou Toujours refuser afin de désactiver l'option Mémoriser pour tous ces sites Web. En d'autre termes, la sélection effectuée ici annule les sélections précédentes effectuées dans le panneau Paramètres de contrôle de l'accès, illustré dans la figure suivante.

Une fois l'option appropriée sélectionnée (Toujours demander ou Toujours refuser), vous pouvez définir les paramètres de contrôle de l'accès pour des sites Web individuels que vous avez déjà consultés. Par exemple, vous pouvez sélectionner Toujours refuser ici, puis utiliser le panneau Paramètres de contrôle de l'accès des sites et sélectionner Toujours autoriser l'accès pour les sites Web individuels que vous connaissez et auxquels vous faites confiance.

Pour le contenu déployé localement et les données locales, les utilisateurs disposent d'une autre option : Ils peuvent spécifier quels fichiers SWF peuvent accéder à Internet en utilisant le panneau Paramètres globaux de sécurité. Pour plus d'informations sur la spécification des paramètres dans le panneau Paramètres globaux de sécurité, consultez la section « Définition des fichiers approuvés en utilisant le Gestionnaire des paramètres », à la page 688. Pour plus d'informations sur le panneau Paramètres globaux de sécurité, consultez le site www.adobe.com/support/documentation/fr/flashplayer/help/settings_manager04a.html.

REMARQUE	Les sélections de l'utilisateur dans le panneau Paramètres globaux de sécurité supplantent toutes les décisions prises dans la boîte de dialogue contextuelle.

Sécurité des fichiers locaux et fichiers de projections

Les fichiers de projections et les fichiers SWF qu'ils contiennent ou chargés dans la projection à l'exécution ne sont pas affectés par les restrictions de sécurité de fichiers locaux, parce que l'utilisateur final doit exécuter l'exécutable afin d'utiliser le fichier SWF. Il n'y a pas de changements des fichiers de projections et de sécurité dans Flash Player 8 et versions ultérieures ; ce programme conserve le même niveau d'accès et de sécurité que dans les versions antérieures.

Rappelez-vous que les utilisateurs sont souvent prudents à propos d'exécuter les fichiers de projections. Un fichier de projections est une application EXE ou Macintosh exécutable, et les utilisateurs doivent être prudents quand il s'agit d'exécuter de tels fichiers sur leurs ordinateurs. Si vous distribuez une application en utilisant des fichiers de projections, certains utilisateurs peuvent ne pas l'installer.

De plus, un fichier de projections incorpore une version spécifique de Flash Player à l'intérieur de la projection, qui peut être antérieure à la dernière version de Flash Player disponible pour le téléchargement à partir du site Web Adobe. Le Flash Player qui est incorporé dans le fichier de projections peut être une version patrimoniale si la projection a été créée avec une version antérieure de Flash ou qu'une édition de Flash Player a été commercialisée après la version actuelle de l'outil de programmation de Flash. Pour ces raisons, vous devez distribuer les applications en utilisant les fichiers SWF quand cela est possible.

Dépannage des fichiers SWF patrimoniaux

Certains fichiers FLA et SWF patrimoniaux (créés avec Flash MX 2004 et antérieur) peuvent ne pas fonctionner lorsque vous les testez ou les déployez localement (sur un disque dur) à cause des modifications de sécurité dans Flash 8 et versions ultérieures. Ceci peut se produire lorsqu'un fichier SWF tente d'accéder aux sites Web en dehors de son domaine, et, dans ce cas, vous devez implémenter un fichier de régulation inter-domaines.

Vous pouvez avoir des fichiers FLA ou SWF créés dans Flash MX 2004 ou antérieur qui ont été distribués à des utilisateurs qui n'utilisent pas l'outil de programmation de Flash mais ont effectué une mise à niveau vers Flash Player 8 et versions ultérieures. Si votre contenu patrimonial testé ou déployé localement (un ancien fichier SWF sur le disque dur d'un utilisateur) s'interrompt parce qu'il tente de communiquer sur Internet lorsqu'il est lu dans Flash Player 8 et versions ultérieures, les utilisateurs devront approuver votre contenu de façon explicite pour qu'il puisse être lu correctement (en cliquant sur un bouton dans la boîte de dialogue).

Pour apprendre comment corriger le contenu patrimonial sur un ordinateur local, consultez la section « Correction du contenu patrimonial déployé sur des ordinateurs locaux », à la page 684.

Correction du contenu patrimonial déployé sur des ordinateurs locaux

Si vous avez publié des fichiers SWF pour Flash Player 7 ou une version antérieure qui sont déployés sur des ordinateurs locaux et communiquent avec Internet, vos utilisateurs devront explicitement les autoriser à communiquer avec Internet. Les utilisateurs peuvent cependant éviter l'interruption du contenu en indiquant l'emplacement du fichier SWF sur leur ordinateur local au Sandbox de confiance dans le Gestionnaire de paramètres.

Pour corriger des fichiers SWF pour la lecture locale utilisez l'une des options suivantes :

Redéployer Exécuter le programme de mise à jour du contenu local. Le programme de mise à jour du contenu local reconfigure votre fichier SWF pour le rendre compatible avec le modèle de sécurité (pour Flash Player 8 et versions ultérieures). Vous reconfigurez le fichier SWF local afin qu'il puisse ou bien accéder uniquement au réseau ou uniquement au système de fichiers local. Pour plus d'informations, et pour télécharger le programme de mise à jour du contenu local, consultez le site www.adobe.com/support/flashplayer/downloads.html.

Republier et redéployer Publier de nouveau le fichier avec Flash. L'outil de programmation nécessite que vous indiquiez dans la boîte de dialogue Paramètres de publication si un fichier SWF local peut accéder au réseau ou au système de fichiers local, mais pas aux deux. Si vous précisez qu'un fichier SWF local peut accéder au réseau, vous devez également activer ses autorisations pour ce fichier SWF (et tous les fichiers SWF locaux) dans les fichiers SWF, HTML, de données et/ou de serveur auxquels il accède. Pour plus d'informations, voir « Publication des fichiers pour le déploiement local », à la page 685.

Déployer un nouveau contenu Utiliser un fichier de configuration (.cfg) dans le dossier #Security/FlashPlayerTrust. Vous pouvez utiliser ce fichier pour définir des autorisations d'accès au réseau et locales. Pour plus d'informations, voir « Création des fichiers de configuration pour le dévelopement Flash », à la page 690.

REMARQUE	Chacune de ces options exige que vous republiez ou redéployiez votre fichier SWF.

Publication des fichiers pour le déploiement local

Vous pouvez envoyer vos fichiers FLA ou SWF Flash à un utilisateur pour tester ou approuver et nécessite que l'application accède à Internet. Si votre document se lit sur un système local mais accède à des fichiers sur Internet (par exemple, chargement XML ou envoi de variables), l'utilisateur peut nécessiter un fichier de configuration pour obtenir un fonctionnement correct, ou il se peut que vous deviez configurer un fichier FLA afin que le fichier SWF que vous publiez puisse accèder au réseau. Sinon, vous pouvez configurer un fichier de configuration à l'intérieur du répertoire FlashPlayerTrust. Pour plus d'informations sur la manière de configurer les fichiers de configuration, consultez la section « Création des fichiers de configuration pour le dévelopement Flash », à la page 690.

Utilisez Flash pour créer le contenu à déployer localement et fonctionnant avec la sécurité de fichiers locaux de Flash Player. Dans les Paramètres de publication de Flash 8 et versions ultérieures, indiquez si le contenu local peut accéder au réseau ou au système de fichiers local, mais pas aux deux.

Vous pouvez définir les niveaux d'autorisation pour un fichier FLA dans la boîte de dialogue Paramètres de publication. Ces niveaux d'autorisation affectent la lecture locale du fichier FLA, lorsqu'il est lu localement sur un disque dur.

REMARQUE

Si vous précisez qu'un fichier SWF local peut accéder au réseau, vous devez également activer les autorisations dans les fichiers SWF, HTML, données et serveur auxquels il accède.

Fichiers SWF réseau Les fichiers SWF qui téléchargent à partir d'un réseau (tel qu'un réseau en ligne) sont placés dans un *Sandbox* séparé qui correspond à leurs domaines d'origine de site Web unique. Les fichiers SWF local qui indiquent qu'ils ont l'accès réseau sont placés dans le Sandbox *local-with-networking*. Par défaut, ces fichiers peuvent lire uniquement les données à partir du site d'origine. La correspondance de domaine exact s'applique à ces fichiers. Les fichiers SWF réseau peuvent accéder aux données à partir d'autres domaines s'ils ont les autorisations appropriées. Pour plus d'informations sur les fichiers SWF réseau, consultez la section « Accès au réseau uniquement », à la page 687.

Fichiers SWF locaux Les fichiers SWF qui fonctionnent avec les systèmes de fichiers locaux ou les chemins réseau UNC sont placés dans un des trois Sandbox dans Flash Player 8 et versions ultérieures. Par défaut, les fichiers SWF locaux sont placés dans le Sandbox *local avec système de fichiers*. Les fichiers SWF locaux qui sont enregistrés comme étant de confiance (en utilisant un fichier de configuration) sont placés dans le Sandbox *local-trusted*. Pour plus d'informations sur les trois Sandbox, consultez la section « Accès aux fichiers locaux uniquement (par défaut) », à la page 686.

Pour plus d'informations sur le Sandbox de sécurité, consultez la section « Fonctionnement des Sandbox de sécurité locale », à la page 679.

Les deux premiers niveaux d'autorisation sont définis dans l'environnement auteur de Flash, et le troisième est défini en utilisant le panneau Paramètres globaux de sécurité ou le fichier FlashAuthor.cfg. L'exemple suivant décrit les options qui sont disponibles lorsque vous publiez un fichier pour le test sur votre disque dur local.

Pour publier un document avec un niveau d'autorisation spécifié :

1. Ouvrez le fichier FLA pour leque vous voulez spécifier un niveau d'autorisation.
2. Choisissez Fichier > Paramètres de publication> Flash.
3. Trouvez la boîte de dialogue Sécurité de lecture locale, et sélectionnez l'une des options suivantes dans le menu contextuel :
   - Accès aux fichiers locaux uniquement (consultez la section « Accès aux fichiers locaux uniquement (par défaut) »)
   - Accès au réseau uniquement (consultez la section « Accès au réseau uniquement »)
4. Cliquez sur OK pour continuer la programmation du fichier FLA, ou cliquez sur Publier pour créer le fichier SWF.

Pour plus d'informations sur les niveaux d'autorisation que vous pouvez définir pour vos applications, consultez « Accès aux fichiers locaux uniquement (par défaut) », à la page 686, « Accès au réseau uniquement », à la page 687, et « Accès au système de fichiers et au réseau », à la page 687.

Accès aux fichiers locaux uniquement (par défaut)

Pour définir le niveau d'autorisation, sélectionnez Paramètres de publication > Flash, et puis sélectionnez Accès aux fichiers locaux uniquement à partir du menu contextuel Sécurité de lecture locale. Ce niveau d'autorisation permet à un fichier SWF local l'accès uniquement au système de fichiers local sur lequel le fichier SWF s'exécute. Le fichier SWF peut lire les fichiers connus sur disque local sans aucune restriction. Cependant, les restrictions suivantes s'appliquent à l'application accédant au réseau :

- Le fichier SWF ne peut en aucune façon accéder au réseau. Le fichier SWF ne peut pas inter-coder les fichiers SWF du réseau, ni être inter-codé par eux.
- Le fichier SWF ne peut pas communiquer avec les fichiers SWF locaux autorisés à accéder au réseau uniquement, ni avec les pages HTML. Cependant, dans certains cas, la communication est permise, comme si le HTML est considéré de confiance et `allowScriptAccess` est défini sur `always` ou si `allowScriptAccess` n'est pas défini et que le fichier SWF est Flash Player 7 ou versions antérieures.

Accès au réseau uniquement

Pour définir ce niveau d'autorisation, sélectionnez Paramètres de publication > Flash, et puis sélectionnez Accès au réseau uniquement à partir du menu contextuel Sécurité de lecture locale. Les fichiers SWF locaux qui ont accès au réseau peuvent lire sur un serveur si celui-ci contient un fichier de régulation inter-domaines avec `<allow-access-from-domain= “*”>`. Les fichiers SWF locaux qui ont accès au réseau peuvent inter-coder d'autres fichiers SWF si ces fichiers accédés contiennent `System.security.allowDomain(“*”)`. Un fichier SWF local avec accès au réseau peut également être inter-codé par les fichiers SWF du réseau s'il contient `allowDomain(“*”)`. Le fichier SWF ne peut jamais lire les fichiers locaux.
Dans certains cas, le type de fichier SWF affecte l'accès. Pour plus d'informations, consultez allowDomain (`méthode security.allowDomain`) dans le *Guide de référence du langage ActionScript 2.0*.

La valeur du caractère générique (*) indique que *tous* les domaines, y compris les hôtes locaux, sont autorisés. Assurez-vous que ce type d'accès est compatible avec votre politique de sécurité avant d'utiliser ce type d'argument.

Sans aucune de ces autorisations, les fichiers SWF locaux avec accès réseau ne peuvent communiquer qu'avec d'autres fichiers SWF locaux avec accès réseau et peuvent envoyer des données aux serveurs (en utilisant `XML.send()`, par exemple). Dans certains cas, l'accès est permit si le fichier HTML est approuvé.

Accès au système de fichiers et au réseau

Ce niveau d'autorisation est le plus élevé. Un fichier SWF local qui dispose de ce type d'autorisation est un *fichier SWF local approuvé*. Les fichiers SWF locaux approuvés peuvent lire les autres fichiers SWF, interagir avec tout serveur, et écrire ActionScript pour d'autres fichiers SWF ou HTML qui ne leur ont pas interdit de façon explicite l'autorisation de fichier (par exemple, avec `allowScriptAccess="none"`). Ce niveau d'autorisation peut être accordé par l'utilisateur ou le développeur Flash des manières suivantes :

- En utilisant le panneau Paramètres globaux de sécurité dans le Gestionnaire des paramètres ;
- En utilisant un fichier de configuration global.

Un fichier de configuration peut être installé avec le fichier SWF, créé par un développeur Flash, ou ajouté par un administrateur (pour tous les utilisateurs ou l'utilisateur actuel) ou tout développeur Flash (pour l'utilisateur actuel).

Pour plus d'informations sur la les fichiers de configuration et le panneau Paramètres globaux de sécurité, consultez « Paramètres de sécurité de Flash Player », à la page 680 et « Définition des fichiers approuvés en utilisant le Gestionnaire des paramètres », à la page 688 et « Création des fichiers de configuration pour le dévelopement Flash », à la page 690.

Test local du contenu avec restrictions de sécurité du fichier Flash local

En tant que développeur Flash, vous testez régulièrement vos applications Flash localement. Lors de ces tests, une boîte de dialogue peut apparaître lorsqu'une application Flash tente de communiquer avec Internet. Cette boîte de dialogue peut apparaître lorsque vous testez un fichier SWF dans Flash Player si le fichier SWF n'a pas accès au réseau. Pour plus d'informations sur la publication de fichiers SWF avec des niveaux d'autorisation spécifiés, consultez « Publication des fichiers pour le déploiement local », à la page 685. Publier un fichier SWF avec une de ces options signifie que vous pouvez communiquer avec le réseau *ou* le système de fichiers local.

Certaines fois, il se peut que vous deviez communiquer avec le système de fichiers local et le réseau lorsque vous testez un document. Parce que le nouveau modèle de sécurité peut interrompre votre flux de travail lorsque vous programmez les applications Flash vous pouvez utiliser le panneau Paramètres globaux de sécurité dans le Gestionnaire des paramètres de Flash Player pour spécifier quelles applications Flash sur votre ordinateur peuvent toujours communiquer avec Internet et le système de fichiers local. Ou, vous pouvez modifier le fichier de configuration pour spécifier les répertoires approuvés sur votre disque dur.

Pour plus d'informations, consultez les sections suivantes :

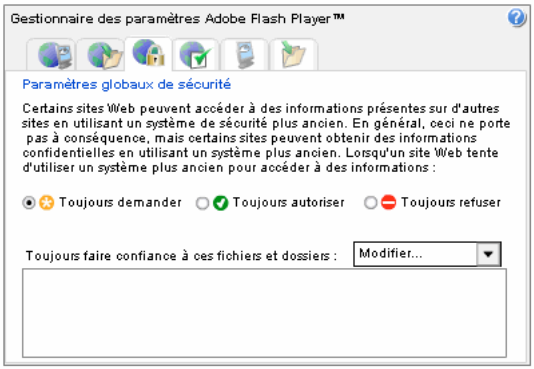
- « Définition des fichiers approuvés en utilisant le Gestionnaire des paramètres », à la page 688
- « Création des fichiers de configuration pour le dévelopement Flash », à la page 690

Définition des fichiers approuvés en utilisant le Gestionnaire des paramètres

Vous pouvez spécifier le contenu Flash auquel votre ordinateur peut systématiquement appliquer les anciennes règles de sécurité en ajoutant l'emplacement de ce contenu dans le panneau Paramètres globaux de sécurité dans le Gestionnaire des paramètres Flash Player. Après avoir ajouté cet emplacement dans le panneau Sécurité de votre ordinateur, son contenu est considéré comme de *confiance*. Flash Player ne vous demandera pas d'autorisation et peut appliquer systématiquement les anciennes règles, même si Toujours refuser est sélectionné dans le panneau Paramètres globaux de sécurité. La liste Faire confiance à cet emplacement est prioritaire par rapport aux options du panneau Paramètres. Ainsi, même si vous avez empêché les contenus locaux et Web d'appliquer les anciennes règles de sécurité, les fichiers locaux figurant dans votre liste de fichiers de confiance pourront continuer à les appliquer.

La liste Faire confiance à cet emplacement située en bas du panneau s'applique uniquement au contenu Flash téléchargé sur votre ordinateur et non pas au contenu utilisé pendant la consultation d'un site Web.

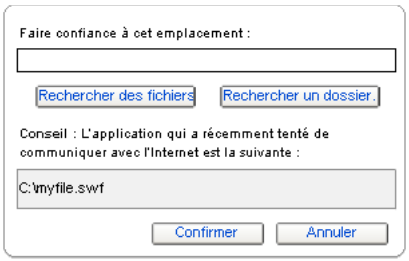
L'exemple suivant décrit comment spécifier qu'un fichier SWF local peut communiquer sur Internet. Lorsque vous testez un fichier dans un navigateur (Fichier > Aperçu avant publication > HTML), une boîte de dialogue de sécurité peut apparaître. Si vous cliquez sur Paramètres, le panneau Paramètres globaux de sécurité du Gestionnaire des paramètres apparaît.



Pour spécifier qu'un fichier SWF local peut communiquer sur Internet et avec le système de fichiers local :

1. Dans le panneau Paramètres globaux de « sécurité », cliquez sur le menu contextuel et sélectionnez Ajouter un emplacement.

   La boîte de dialogue correspondante s'affiche.



   Si vous avez accédé au Gestionnaire de paramètres en cliquant sur le bouton Paramètres d'une boîte de dialogue, la boîte de dialogue Ajouter un emplacement contient un chemin de type C:*nomrépertoire\nomfichier.swf* ou */Utilisateurs/nomRépertoire/nomFichier.swf*. Ce chemin indique le fichier qui a tenté de communiquer sur Internet et a été bloqué par la fonction de sécurité de Flash Player. Si ce chemin contient le contenu dont vous souhaitez effectivement autoriser la communication sur Internet, copiez et collez-le dans la zone Faire confiance à cet emplacement. Vous pouvez également cliquer sur le bouton Parcourir et sélectionner le contenu manuellement.

Vous pouvez ajouter aussi bien un fichier qu'un répertoire. Si vous ajoutez un répertoire, tous les fichiers et les sous-répertoires de ce dernier sont considérés comme de confiance. Certains contenus Flash consistent en plusieurs fichiers liés entre eux. Dans ce cas, il est sans doute préférable de faire confiance au répertoire contenant ces fichiers. De manière générale, évitez de faire confiance aux répertoires de niveau supérieur.

2. Cliquez sur Confirmer.

   L'emplacement est ajouté au panneau des paramètres de sécurité. Les emplacements figurant dans la liste sont toujours autorisés à appliquer les anciennes règles de sécurité, même si les options Toujours refuser ou Toujours demander, situées dans la partie supérieure du panneau Paramètres globaux de sécurité, sont sélectionnées.

   Après avoir sélectionné les emplacements de confiance, vous devez redémarrer le contenu Flash en actualisant le navigateur ou en redémarrant Flash Player.

Si vous cliquez sur Toujours autoriser, ceci applique uniquement ce paramètre pour toujours autoriser le contenu patrimonial (Flash Player 7 et versions antérieures). Le paramètre n'autorise pas toujours le contenu Flash Player 8 et versions ultérieures. Nous vous conseillons par contre de spécifier quelles applications et quels répertoires Flash de votre ordinateur sont autorisés à communiquer avec Internet et le système de fichiers local.

Création des fichiers de configuration pour le dévelopement Flash

L'outil de programmation de Flash définit un indicateur sur votre disque dur qui vous identifie en tant que développeur et vous dirige vers une version destinée aux développeurs spécifique du panneau Paramètres globaux de sécurité plutôt que vers une destinée aux utilisateurs. Cet indicateur se trouve dans le fichier FlashAuthor.cfg sur votre disque dur, copié automatiquement lors de l'installation de l'outil de programmation de Flash.

Le fichier FlashAuthor.cfg est situé dans les répertoires proches suivants :

Windows *boot disk*\Documents and Settings\<*NomUtilisateur*>\Application Data\Adobe\Flash Player\#Security

Macintosh /Utilisateurs/<*NomUtilisateur*>/Library/Preferences/Adobe/Flash Player/#Security/

Par défaut, ce fichier est défini à `LocalSecurityPrompt=Author`, ce qui signifie que les avertissements que vous voyez sur votre ordinateur vous traitent en tant que développeur Flash contrairement à un utilisateur sans outil de programmation installé.

Vous pouvez tester vos applications locales en tant qu'utilisateur final et voir les boîtes de dialogue d'avertissements qu'il pourrait rencontrer. Pour ce faire, ouvrez FlashAuthor.cfg dans un éditeur de texte, et modifiez LocalSecurityPrompt dans le fichier FlashAuthor.cfg pour correspondre à ce qui suit :

```
LocalSecurityPrompt=User
```

Vous pouvez vouloir fournir un fichier FlashAuthor.cfg, avec `LocalSecurityPrompt` défini sur `Author`, pour d'autres développeurs ou processus de développement ou aux utilisateurs qui testent les applications Flash sur leur disque dur local et n'ont pas l'outil de programmation de Flash installé. Ceci vous aide à reproduire l'expérience de l'utilisateur final avec votre contenu déployé localement.

REMARQUE	Si le fichier FlashAuthor.cfg est supprimé, le fichier est recréé lorsque vous lancez l'outil de programmation de Flash.

Dans le répertoire #Security sur votre disque dur, vous pouvez créer un répertoire FlashPlayerTrust dans lequel vous pouvez stocker des fichiers de configuration uniques. Dans ces fichiers, vous pouvez spécifier les répertoires ou applications de confiance sur votre disque dur. Ce répertoire ne nécessite pas d'accès administratif, alors les utilisateurs sans autorisation administrative peuvent définir des autorisations pour les fichiers SWF et tester les applications.

Si vous ne spécifiez pas un répertoire, votre contenu peut ne pas fonctionner comme prévu. Les fichiers de configuration dans un répertoire FlashPlayerTrust contiennent des chemins de répertoires. Le fichier peut contenir une liste de plusieurs répertoires, et vous pouvez ajouter de nouveaux chemins au fichier. Flash Player s'attend à un chemin par ligne dans les fichiers de configuration. Toute ligne qui commence par un ponctuateur # (sans espace avant) est traitée comme un commentaire.

Pour créer un fichier de configuration qui fait confiance à un répertoire :

1. Localisez le dossier #Security sur votre disque dur.
2. Créez un dossier appelé **FlashPlayerTrust** dans le dossier #Security.
3. Créez un nouveau fichier appelé dans le répertoire FlashPlayerTrust en utilisant un éditeur de texte, et enregistrez-le sous **myTrustFiles.cfg**.

   Vous pouvez utiliser n'importe quel nom unique pour votre fichier de configuration.
4. Localisez le répertoire dans lequel vous testez les applications Flash.

5. Saisissez ou collez chaque chemin de répertoire (tout chemin de répertoire sur votre disque dur) sur une nouvelle ligne dans le fichier. Vous pouvez coller des chemins de répertoires multiples sur des lignes séparées. Une fois que vous avez terminé, votre fichier ressemble à l'exemple suivant :

```
C:\Documents and Settings\<votrenom>\My Documents\files\
C:\Documents and Settings\<votrenom>\My Documents\testapps\
```

6. Enregistrez les modifications apportées à myTrustFiles.cfg.
7. Testez un document qui accède aux fichiers locaux et de réseau à partir du répertoire que vous avez ajouté au fichier.

   Les applications Flash enregistrées dans ce répertoire peuvent maintenant accéder aux fichiers locaux et au réseau.

Il peut y avoir de nombreux chemins de répertoires enregistrés dans chaque fichier de configuration, et de nombreux fichiers *.cfg enregistrés dans le répertoire FlashPlayerTrust.

Si vous créez des applications qui s'installent sur un disque dur d'utilisateur final, vous pouvez avoir besoin de créer un fichier de configuration dans FlashPlayerTrust afin de spécifier un répertoire approuvé pour votre application. Vous pouvez créer des fichiers de configuration dans le répertoire FlashPlayerTrust qui spécifient l'emplacement de l'application approuvée. Consultez la procédure précédente pour les informations sur ce répertoire et la création de fichiers de configuration.

REMARQUE	Une installeur est exécuter par un utilisateur avec une autorisation administrative sur un ordinateur.

Vous devez développer un schéma unique d'attribution de noms pour empêcher les conflits avec d'autres applications qui peuvent installer des fichiers dans ce répertoire. Par exemple, vous pouvez vouloir utiliser le nom unique de votre société et logiciel dans le nom du fichier pour empêcher les conflits.

CONSEIL	Si vous ne voulez pas utiliser les fichiers de configuration, vous pouvez publier vos applications Flash sur un serveur de test distinct au lieu de donner des fichiers SWF aux clients ou autres développeurs à exécuter sur leurs disques durs locaux.

Pour plus d'informations sur les fichiers de configuration, consultez le site www.adobe.com/go/flashauthorcfg_fr. Vous pouvez aussi créer un fichier de configuration unique pour approuver un ou plusieurs répertoires. Pour des informations détaillées sur la sécurité, consultez le site www.adobe.com/devnet/security/ et www.adobe.com/software/flashplayer/security/.

A propos de la propriété sandboxType

La propriété `System.security.sandboxType` de Flash Player 8 et versions ultérieures retourne le type de Sandbox de sécurité dans lequel le fichier SWF d'appel fonctionne.

La propriété `sandboxType` peut avoir l'une des quatre valeurs suivantes :

remote Le SWF est hébergé sur Internet et fonctionne sous les règles Sandbox selon le domaine.

localTrusted Le fichier SWF est un fichier local qui a été approuvé par l'utilisateur, en utilisant ou bien le Gestionnaire de paramètres globaux de sécurité ou un fichier de configuration FlashPlayerTrust. Le fichier SWF peut lire les sources de données locales et communiquer sur le réseau (tel qu'Internet).

localWithFile Le fichier SWF est un fichier local qui n'a pas été approuvé par l'utilisateur, et n'a pas été publié avec la désignation de mise en réseau. Le fichier SWF peut lire les sources de données locales mais ne peut pas communiquer sur le réseau (tel qu'Internet).

localWithNetwork Le fichier SWF est un fichier local qui n'a pas été approuvé par l'utilisateur, et a été publié avec Accès au réseau uniquement sélectionné dans la boîte de dialogue Paramètres de publication (onglet Flash). Le fichier SWF peut communiquer sur le réseau mais ne peut pas lire les sources de données locales.

Vous pouvez vérifier la propriété `sandboxType` à partir de n'importe quel fichier SWF, bien qu'une valeur soit renvoyée uniquement dans les fichiers publiés pour Flash Player 8 et versions ultérieures. Ceci signifie que lorsque vous publiez pour Flash Player 7 ou les versions antérieures, vous ne savez pas si la propriété `sandboxType` est prise en charge à l'exécution. Si la propriété n'est pas prise en charge à l'exécution, la valeur est `undefined`, ce qui se produit lorsque la version de Flash Player (indiquée par la propriété `System.capabilities.version`) est antérieure à 8. Si la valeur est `non définie`, vous pouvez déterminer le type de Sandbox selon si l'URL du fichier SWF est un fichier local ou non. Si le fichier SWF est un fichier local, Flash Player classe votre fichier SWF en tant que `localTrusted` (ce qui est comment tout le contenu local a été traité avant Flash Player 8); autrement Flash Player classe le fichier SWF en tant que `remote`.

A propos des restrictions Local-with-file-system

Un fichier local-with-file-system n'a pas été enregistré en utilisant le fichier de configuration dans le répertoire FlashPlayerTrust, le panneau Paramètres globaux de sécurité dans le Gestionnaire des paramètres, ou n'a pas été autorisé pour le réseau dans la boîte de dialogue Paramètres de publication dans l'environnement auteur de Flash.

REMARQUE

Pour plus d'informations sur les Sandbox de sécurité, consultez la section « Fonctionnement des Sandbox de sécurité locale », à la page 679.

Ces fichiers comprennent le contenu patrimonial qui peut être lu par Flash Player 8 et versions ultérieures. Si vous développez du contenu dans Flash 9 ou que vous avez du contenu qui se range dans une des catégories suivantes, vous (ou vos utilisateurs) devriez enregistrer le fichier comme approuvé. Pour plus d'informations sur l'enregistrement d'un fichier comme approuvé, consultez la section « Définition des fichiers approuvés en utilisant le Gestionnaire des paramètres », à la page 688. Pour plus d'informations sur l'autorisation pour la lecture de fichier local en utilisant les fichiers de configuration, consultez « Création des fichiers de configuration pour le dévelopement Flash », à la page 690.

Les fichiers SWF Local-with-file-system ont les restrictions suivantes :

- Ne peuvent pas accéder au réseau, ce qui comprend ce qui suit :
  - Charger d'autres fichiers SWF à partir du réseau (*sauf* en utilisant les chemins UNC non Internet)
  - Envoyer des requêtes HTTP
  - Effectuer des connexions en utilisant XMLSocket, Flash Remoting ou NetConnection
  - Appeler `getURL()` *sauf* si vous utilisez `getURL("file:...")` ou `getURL("mailto:...")`
- Peuvent interagir avec d'autres fichiers Local-with-file-system, mais comprennent des restrictions pour ce qui suit :
  - Codage croisé (tel que l'accès ActionScript aux objets dans d'autres fichiers SWF).
  - Appeler `System.security.allowDomain`
  - Utiliser `LocalConnection` comme expéditeur ou écouteur et quels que soient les gestionnaires `LocalConnection.allowDomain`.

REMARQUE

Les fichiers Local-with-file-system peuvent interagir avec d'autres fichiers local-with-file-system, fichiers SWF non réseau. Cependant, ils ne peuvent pas interagir avec les fichiers SWF local-with-network.

Les fichiers SWF Local-with-file-system ont l'accès de lecture aux fichiers connus sur le système de fichiers local. Par exemple, vous pouvez utiliser `XML.load()` dans un fichier SWF local-with-file-system aussi longtemps que vous chargez à partir du sytème de fichiers local et non pas d'Internet.

- Les fichiers SWF Local-with-file-system ne peuvent pas communiquer avec les pages HTML, ce qui comprend ce qui suit :
  - Codage entrant (tel que ExternalInterface API, ActiveX, LiveConnect, et XPConnect)
  - Codage sortant (tel que les appels personnalisés `fscommand` et `getURL("javascript:...")`)

REMARQUE	L'une des exceptions à ceci est si la page HTML est autorisée.

Restriction des API de réseau

Vous pouvez contrôler l'accès aux fonctionnalités réseau d'un fichier SWF en définissant le paramètre `allowNetworking` des balises `<object>` et `<embed>` de la page HTML contenant le fichier SWF.

Les valeurs possibles de `allowNetworking` sont les suivantes :

- `"all"` (valeur par défaut) — Toutes les API de réseau sont autorisées dans le fichier SWF.
- `"internal"` — Le fichier SWF ne peut pas appeler les API de navigation du navigateur ou les API d'interaction du navigateur, énumérées ci-dessous, mais il peut appeler toutes les autres API de réseau.
- `"none"` — Le fichier SWF ne peut appeler aucune des API de réseau, énumérées ci-dessous. Il ne peut également pas utiliser d'API de communication SWF à SWF, figurant également dans la liste ci-dessous.

Pour définir le paramètre `allowNetworking`, dans les balises `<object>` et `<embed>` de la page HTML contenant le fichier SWF, ajoutez le paramètre `allowNetworking` et définissez sa valeur, comme dans l'exemple suivant :

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
 codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/
  swflash.cab#version=9,0,18,0"
 width="600"  height="400" id="test" align="middle">
<param name="allowNetworking" value="none" />
<param name="movie" value="test.swf" />
<param name="bgcolor" value="#333333" />
<embed src="test.swf" allowNetworking="none" bgcolor="#333333" width="600"
  height="400"
 name="test" align="middle" type="application/x-shockwave-flash"
 pluginspage="http://www.adobe.com/shockwave/download/
  download.cgi?P1_Prod_Version=ShockwaveFlash" />
</object>
```

Les API suivantes ne sont pas autorisées lorsque `allowNetworking` est défini sur `"internal"`:

- `getURL`
- `MovieClip.getURL`
- `fscommand()`
- `ExternalInterface.call()`

Outre les API précédentes, les API suivantes ne sont pas autorisées lorsque `allowNetworking` est défini sur `"none"`:

- `XML.load`
- `XML.send`
- `XML.sendAndLoad`
- `LoadVars.load()`
- `LoadVars.send`
- `LoadVars.sendAndLoad`
- `loadVariables`
- `loadVariablesNum`
- `MovieClip.loadVariables`
- `NetConnection.connect`
- `NetStream.play`
- `loadMovie`
- `loadMovieNum`
- `MovieClip.loadMovie`
- `MovieClipLoader.loadClip`
- `Sound.loadSound`
- `LocalConnection.connect`
- `LocalConnection.send`
- `SharedObject.getLocal`
- `SharedObject.getRemote`
- `FileReference.upload`
- `FileReference.download`
- `System.security.loadPolicyFile`
- `XMLSocket.connect`

Même si le réglage `allowNetworking` sélectionné autorise un fichier SWF à utiliser une API de réseau, il peut exister d'autres restrictions basées sur des limitations de sandbox de securité, comme décrit dans le présent chapitre.

Si `allowNetworking` est défini sur `"none"`, les balises `<img>` figurant dans la propriété `htmlText` d'un objet TextField ne s'affichent pas dans le contenu réseau.Si `allowNetworking` est défini sur `"none"`, les symboles d'une bibliothèque partagée importée ajoutés dans l'outil de programmation de Flash (et non ActionScript) ne sont pas autorisés lors de l'exécution.

Domaines, sécurité inter-domaines et fichiers SWF

Par défaut, Flash Player 7 et ses versions ultérieures empêchent tout fichier SWF servi par un domaine de lire les données, objets ou variables de fichiers SWF servis par des domaines différents. En outre, le contenu chargé à l'aide de protocoles non sécurisés (non-HTTPS) ne peut pas lire le contenu chargé à l'aide d'un protocole sécurisé (HTTPS), même s'ils sont tous les deux situés exactement dans le même domaine. Par exemple, un fichier SWF situé à http://www.adobe.com/main.swf ne peut pas charger les données de https://www.adobe.com/data.txt sans autorisation explicite. De même, un fichier SWF servi à partir d'un domaine ne peut pas charger des données (avec `loadVars()`, par exemple) à partir d'un autre domaine.

Les adresses IP numériques identiques sont compatibles. Cependant, un nom de domaine n'est pas compatible avec une adresse IP, même s'il renvoie à la même adresse IP.

Le tableau suivant présente des exemples de domaines compatibles :

www.adobe.com	www.adobe.com
data.adobe.com	data.adobe.com
65.57.83.12	65.57.83.12

Le tableau suivant présente des exemples de domaines incompatibles :

www.adobe.com	data.adobe.com
adobe.com	www.adobe.com
www.adobe.com	adobe.com
65.57.83.12	www.adobe.com (même si ce domaine correspond à l'adresse 65.57.83.12)
www.adobe.com	65.57.83.12 (même si www.adobe.com correspond à cette adresse IP)

Flash Player 8 et versions ultérieures ne permettent pas aux fichiers SWF locaux de communiquer avec Internet sans une configuration appropriée. Pour plus d'informations sur le paramétrage des fichiers de configuration, consultez la section « Création des fichiers de configuration pour le dévelopement Flash », à la page 690.

Pour plus d'informations sur la sécurité, consultez le site www.adobe.com/devnet/security/ et www.adobe.com/software/flashplayer/security/.

Pour plus d'informations, voir les sections suivantes :

- « Règles de nom de domaine pour les param¨ètres et les données locaux », à la page 698
- « Accès inter-domaines et accès aux sous-domaines entre fichiers SWF », à la page 699
- « Autorisation de chargement de données inter-domaines », à la page 706

Règles de nom de domaine pour les param¨ètres et les données locaux

Dans Flash Player 6, les règles de correspondance de superdomaine sont utilisées par défaut pour accéder aux paramètres locaux (tels que les autorisations d'accès de la caméra ou du microphone) ou aux données persistantes localement (objets partagés). Cela signifie que les paramètres et données de fichiers SWF résidant sur one.adobe.com, two.adobe.com et adobe.com sont partagés et tous enregistrés sur adobe.com.

Dans Flash Player 7, les règles de correspondance exacte de domaine sont utilisées par défaut. Cela signifie que les paramètres et données d'un fichier hébergé sur one.adobe.com sont enregistrés sur one.adobe.com, les paramètres et données d'un fichier hébergé sur two.adobe.com sont enregistrés sur two.adobe.com et ainsi de suite. `System.exactSettings`, vous permet de spécifier les règles à utiliser. Cette propriété est prise en charge pour les fichiers publiés pour Flash Player 6 ou une version ultérieure. Pour les fichiers publiés pour Flash Player 6, la valeur par défaut est `false`, ce qui signifie que les règles de correspondance de superdomaine sont utilisées. Pour les fichiers publiés pour Flash Player 7 ou versions ultérieures, la valeur par défaut est `true`, ce qui signifie que les règles de correspondance exacte de domaine sont utilisées. Si vous utilisez des paramètres ou des données locales persistantes et souhaitez publier un fichier SWF Flash Player 6 pour Flash Player 7 ou versions ultérieures, il peut être nécessaire de définir cette valeur sur `false` dans le fichier patrimonial. Pour plus d'informations, consultez `exactSettings (propriété System.exactSettings)` dans le *Guide de référence du langage ActionScript 2.0*.

Accès inter-domaines et accès aux sous-domaines entre fichiers SWF

Lorsque vous développez une série de fichiers SWF qui communiquent entre eux en ligne (par exemple, lorsque vous utilisez `loadMovie()`, `MovieClip.loadMovie()`, `MovieClipLoader.LoadClip()` ou des objets Local Connection), vous pouvez les héberger dans des domaines différents ou dans des sous-domaines différents d'un même superdomaine.

Dans les fichiers publiés pour Flash Player 5 ou une version antérieure, l'accès inter-domaines et aux sous-domaines n'était soumis à aucune restriction.

Dans les fichiers publiés pour Flash Player 6, vous pouviez utiliser le gestionnaire `LocalConnection.allowDomain` ou la méthode `System.security.allowDomain()` pour autoriser l'accès inter-domaines (par exemple pour l'accès à un fichier situé sur adobe.com par un fichier situé sur helpexamples.com). Aucune commande n'était nécessaire pour autoriser l'accès au superdomaine (par exemple, un fichier situé sur www.adobe.com était accessible par un fichier situé sur something.adobe.com).

Les fichiers publiés pour Flash Player 7 implémentent l'accès entre les fichiers SWF différemment des versions précédentes, et ce de deux manières : tout d'abord, Flash Player 7 implémente les règles de correspondance exacte de domaine et non celles de superdomaine. Ainsi, le fichier accédé (même s'il est publié pour une version antérieure à Flash Player 7) doit autoriser de façon explicite l'accès inter-domaines et aux sous-domaines. Ce sujet est abordé dans cette section. Ensuite, un fichier hébergé sur un site utilisant un protocole sécurisé (HTTPS) doit autoriser de façon explicite l'accès à partir d'un fichier hébergé sur un site utilisant un protocole non sécurisé (HTTP ou FTP). Ce sujet est abordé dans la section suivante (consultez la section « Accès du protocole HTTP vers le protocole HTTPS entre les fichiers SWF », à la page 711).

Vous appelez généralement `System.security.allowDomain` dans vos applications. Cependant, lorsque le récepteur LocalConnection est un fichier SWF HTTPS et que l'expéditeur ne l'est pas, `allowInsecureDomain` est appelé à la place.

Le problème suivant affecte uniquement les fichiers SWF publiés pour Flash Player 7. Lorsque le récepteur est HTTPS et que l'expéditeur est un fichier SWF local, `allowDomain()` est appelé, même si `allowInsecureDomain()` devrait être appelé. Cependant, dans Flash Player 8 et versions ultérieures, lorsque le récepteur LocalConnection HTTPS est Flash Player 8 et versions ultérieures et que l'expéditeur est un fichier local, `allowInsecureDomain()` est appelé.

Les fichiers qui s'exécutent dans Flash Player 8 et versions ultérieures sont sujets à des modification par rapport à la façon dont ils s'exécutent dans Flash Player 7. Appeler `System.security.allowDomain` permet les opérations d'inter-codage uniquement où le fichier SWF accèdé est celui qui a appelé `System.security.allowDomain`. En d'autres mots, un fichier SWF qui appelle `System.security.allowDomain` autorise maintenant uniquement l'accès à lui- même. Dans les versions précédentes, l'appel de `System.security.allowDomain` autorisait les opérations de programmation croisée lorsque le fichier SWF cible appartenait au même domaine que le fichier SWF qui a appelé `System.security.allowDomain`. Ceci a ouvert tout le domaine du fichier SWF appelant.

La prise en charge a été ajoutée pour la valeur du caractère générique (*) à `System.security.allowDomain("*")` et `System.security.allowInsecureDomain("*")`. La valeur caractère générique (*) autorise les opérations de programmation croisée quel que soit le fichier procédant à l'accès et quel que soit l'emplacement de ce dernier (tel qu'une autorisation globale). Les autorisations de caractère génériques peuvent être utiles, mais elles doivent respecter les nouvelles règles de sécurité de fichiers locaux dans Flash Player 8 et versions ultérieures. Particulièrement, les fichiers locaux ne proviennent pas d'un domaine, alors la valeur de caractère générique doit être utilisée. Cependant, soyez prudent lors de l'utilisation de la valeur de caractère générique parce que n'importe quel domaine a accès à votre fichier. Pour plus d'informations, consultez `allowInsecureDomain (méthode security.allowInsecureDomain)`.

Vous pouvez rencontrer une situation lorsque vous chargez un fichier SWF enfant à partir d'un domaine différent de celui qui l'appelle. Vous pouvez souhaiter permettre que ce fichier code le fichier SWF parent, mais vous ne connaissez pas le domaine final à partir duquel est issu le fichier SWF enfant. Cela peut se produire, par exemple, lorsque vous utilisez des redirections d'équilibrage de charge ou des serveurs tiers. Dans ce cas, vous pouvez utiliser la propriété `MovieClip._url` comme argument de cette méthode. Par exemple, si vous chargez un fichier SWF dans `my_mc`, vous pouvez appeler `System.security.allowDomain(my_mc._url)`. Si vous procédez ainsi, veuillez patienter jusqu'au début du chargement du fichier SWF dans `my_mc` car la propriété `_url` ne dispose de sa valeur correcte et finale qu'à ce moment là. Afin de déterminer si le chargement d'un fichier SWF enfant a commencé, utilisez `MovieClipLoader.onLoadStart`.

La situation opposée peut également se produire ; en effet, vous pouvez créer un fichier SWF enfant sur lequel son fichier parent pourra créer un script, mais qui ignore le domaine de celui-ci (ce qui signifie, c'est un fichier SWF qui peut être chargé par une variété de domaines). Dans ce cas, appelez `System.security.allowDomain(_parent._url)` à partir du fichier SWF enfant. Il n'est pas nécessaire d'attendre la fin du chargement du fichier SWF parent parce qu'il sera déjà chargé lorsque celui de l'enfant commencera.

REMARQUE

(Si le fichier SWF Internet cible est chargé à partir d'une URL HTTPS, le fichier SWF Internet doit appeler `System.security.allowInsecureDomain("*")`.

Le tableau suivant récapitule les règles de correspondance des domaines dans les différentes versions de Flash Player.

Fichiers publiés pour Flash Player	Accès inter-domaines entre fichiers SWF (allowDomain() est nécessaire)	Accès aux sous-domaines entre fichiers SWF
5 ou plus récente	Aucune restriction	Aucune restriction
6	Correspondance de superdomaine : `allowDomain()` est nécessaire si les superdomaines ne concordent pas	Aucune restriction
7 et version ultérieure	Correspondance exacte des domaines Autorisation explicite d'accès aux fichiers hébergés sur des sites HTTP ou FTP pour les fichiers HTTPS	Correspondance exacte des domaines Autorisation explicite d'accès aux fichiers hébergés sur des sites HTTP ou FTP pour les fichiers HTTPS

REMARQUE

Vous avez besoin de `System.security.allowInsecureDomain` dans Flash Player 7 et ultérieur si vous effectuez un accès HTTP-à-HTTPS, même si vous avez une correspondance de domaines exacte.

Les versions qui contrôlent le comportement de Flash Player sont les versions des fichiers SWF (la version Flash Player spécifiée d'un fichier SWF), non pas la version de Flash Player. Par exemple, lorsque Flash Player 8 et ses versions ultérieures lit un fichier SWF publié pour la version 7, Flash Player applique les comportements correspondant à cette version. Cette pratique permet de s'assurer que les mises à niveau du lecteur ne changent pas le comportement de `System.security.allowDomain` dans les fichiers SWF déployés.

Flash Player 7 et ultérieur implémentant les règles de correspondance exacte de domaine et non les règles de correspondance de superdomaine, il peut être nécessaire de modifier les scripts existants si vous souhaitez les lire à partir de fichiers publiés pour Flash Player 7 et ses versions ultérieures. (Vous pouvez toujours publier les fichiers modifiés pour Flash Player 6.) Si vous avez utilisé une instruction `LocalConnection.allowDomain()` ou `System.security.allowDomain()` dans vos fichiers et que vous avez autorisé des sites de superdomaine, vous devez modifier vos paramètres pour spécifier les domaines exacts en remplacement. L'exemple suivant présente des modifications qui peuvent être nécessaires si vous avez le code Flash Player 6 :

```
// Commandes Flash Player 6 d'un fichier SWF situé sur www.helpexamples.com
// pour autoriser l'accès par des fichiers SWF hébergés sur www.adobe.com
// ou sur store.adobe.com
System.security.allowDomain("adobe.com");
ma_lc.allowDomain = function(sendingDomain) {
  return(sendingDomain=="adobe.com");
}
// Commandes correspondantes pour autoriser l'accès par les fichiers SWF
// qui sont publiés pour Flash Player 7 ou ultérieur
System.security.allowDomain("www.adobe.com", "store.adobe.com");
ma_lc.allowDomain = function(sendingDomain) {
  return(sendingDomain=="www.adobe.com" ||
    sendingDomain=="store.adobe.com");
}
```

Il peut également être nécessaire d'ajouter des instructions similaires dans vos fichiers si vous ne les utilisez pas actuellement. Par exemple, si votre fichier SWF est hébergé sur www.adobe.com et que vous souhaitez autoriser l'accès par un fichier SWF publié pour Flash Player 7 ou ultérieur sur store.adobe.com, vous devez ajouter des instructions semblables à l'exemple suivant au fichier situé sur www.adobe.com (vous pouvez toujours publier le fichier situé sur www.adobe.com pour Flash Player 6) :

```
System.security.allowDomain("store.adobe.com");
ma_lc.allowDomain = function(sendingDomain) {
  return(sendingDomain=="store.adobe.com");
}
```

En outre, si une application Flash Player 6 qui s'exécute sous Flash Player 7 tente d'accéder à des données en dehors de son domaine exact, les règles de correspondance de domaine de Flash Player 7 et ultérieur s'appliquent et l'utilisateur doit autoriser ou refuser l'accès.

Pour résumer, il peut être nécessaire de modifier vos fichiers pour ajouter ou modifier des instructions `allowDomain` si vous publiez des fichiers pour Flash Player 7 ou ultérieur dans les conditions suivantes :

- Vous avez implémenté le codage de fichiers inter-SWF (consultez « Autorisation d'accès aux données entre fichiers SWF inter-domaines », à la page 703).

- Le fichier SWF appelé (quelle que soit sa version) n'est pas hébergé sur un site utilisant un protocole sécurisé (HTTPS), ou les fichiers SWF appelé et appelant sont tous deux hébergés sur des sites sécurisés (HTTPS). (Si seul le fichier SWF appelé est sécurisé, consultez la section « Accès du protocole HTTP vers le protocole HTTPS entre les fichiers SWF », à la page 711).
- Les fichiers SWF ne se trouvent pas dans le même domaine (par exemple, l'un d'eux réside sur www.adobe.com et l'autre sur store.adobe.com).

Vous devez effectuer les modifications suivantes :

- Si le fichier SWF appelé est publié pour Flash Player 7ou ultérieur, incluez-y `System.security.allowDomain` ou `LocalConnection.allowDomain` en utilisant la correspondance exacte de domaine.
- Si le fichier SWF appelé est publié pour Flash Player 6, modifiez-le en ajoutant ou en modifiant une instruction `System.security.allowDomain` ou `LocalConnection.allowDomain`, en utilisant la correspondance exacte de domaine, comme dans les exemples de code précédents de cette section. Vous pouvez publier le fichier modifié pour Flash Player 6 ou 7.
- Si le fichier SWF appelé est publié pour Flash Player 5 ou une version antérieure, exportez-le dans Flash Player 6 ou 7 et ajoutez-lui une instruction `System.security.allowDomain` en utilisant la correspondance exacte de domaine, comme dans les exemples de code précédents de cette section. (Les objets LocalConnection ne sont pas pris en charge par Flash Player 5 ou version antérieure.)

Pour plus d'informations sur les Sandbox de sécurité, consultez la section « Sécurité des fichiers locaux et Flash Player », à la page 677.

Autorisation d'accès aux données entre fichiers SWF inter-domaines

Pour que chacun des deux fichiers SWF puisse accéder aux données (variables et objets) de l'autre, ils doivent provenir du même domaine. Par défaut, dans Flash Player 7 et versions ultérieures, les deux domaines doivent correspondre exactement pour que les deux fichiers partagent des données. Un fichier SWF peut cependant accorder l'accès aux fichiers SWF servis par des domaines spécifiques en appelant `LocalConnection.allowDomain` ou `System.security.allowDomain()`.

`System.security.allowDomain()` permet aux fichiers SWF et HTML dans des domaines spécifiés l'accès aux objets et variables dans le fichier SWF qui contient l'appel `allowDomain()`.

Si deux fichiers SWF sont servis à partir du même domaine, par exemple, http://adobe.com/movieA.swf et http://adobe.com/movieB.swf, alors movieA.swf peut alors analyser et modifier les variables, les objets, les propriétés, les méthodes, etc. dans movieB.swf, et movieB peut faire la même chose pour movieA. Ceci est appelé programmation entre plusieurs animations ou *programmation croisée*.

Si deux fichiers SWF sont servis à partir de différents domaines, par exemple http://adobe.com/movieA.swf et http://helpexamples.com/movieB.swf, puis, par défaut, Flash Player ne permet pas à movieA.swf de programmer movieB.swf, ou à movieB de programmer movieA. Si vous appelez `System.security.allowDomain("adobe.com")`, movieB.swf accorde à movieA.swf l'autorisation de programmer movieB.swf. Un fichier SWF donne aux fichiers SWF provenant d'autres domaines l'autorisation de le programmer en appelant `System.security.allowDomain()`. Ceci s'appelle programmation de *scripts interdomaine*.

Pour plus d'informations sur `System.security.allowDomain()`, inter-codage et codage inter-domaines, consultez la section allowDomain (méthode security.allowDomain) dans le *Guide de référence du langage ActionScript 2.0*.

Par exemple, supposons que le fichier main.swf provienne de www.adobe.com. Il charge alors un autre fichier SWF (data.swf) à partir de data.adobe.com vers une occurrence de clip créée de façon dynamique avec`createEmptyMovieClip()`.

```
// Dans adobe.swf
this.createEmptyMovieClip("target_mc", this.getNextHighestDepth());
target_mc.loadMovie("http://data.adobe.com/data.swf");
```

Supposons que le fichier data.swf définisse une méthode appelée `getData()` sur son scénario principal. Par défaut, main.swf ne peut pas appeler la méthode `getData()` définie dans data.swf après que ce fichier a été chargé, car ces deux fichiers SWF ne résident pas dans le même domaine. En ce sens, l'appel de méthode suivant dans main.swf, une fois le chargement de data.swf effectué, échoue :

```
// Dans adobe.swf, après le chargement de data.swf :
target_mc.getData(); // Cette méthode d'appel va échouer
```

Cependant, le fichier data.swf peut accéder à des fichiers SWF servis à partir de www.adobe.com à l'aide du gestionnaire `LocalConnection.allowDomain` et de la méthode `System.security.allowDomain()`, selon le type d'accès nécessaire. Le code suivant, ajouté au fichier data.swf, permet à un fichier SWF servi à partir de www.adobe.com d'accéder à ses variables et méthodes :

```
// Dans data.swf
this._lockroot = true;
System.security.allowDomain("www.adobe.com");
var my_lc:LocalConnection = new LocalConnection();
my_lc.allowDomain = function(sendingDomain:String):Boolean {
  return (sendingDomain == "www.adobe.com");
};
function getData():Void {
  var timestamp:Date = new Date();
  output_txt.text += "data.swf:" + timestamp.toString() + "\n\n";
}
output_txt.text = "**INIT**:\n\n";
```

Désormais, la fonction `getData` du fichier SWF chargé peut être appelée par le fichier adobe.swf. Notez que la commande `allowDomain` permet à tout fichier SWF du domaine autorisé de programmer tout autre fichier SWF dans le domaine qui autorise l'accès, à moins que le fichier SWF exploité soit hébergé sur un site utilisant un protocole sécurisé (HTTPS).

Pour plus d'informations sur la mise en correspondance des noms de domaines, consultez la section « Accès inter-domaines et accès aux sous-domaines entre fichiers SWF », à la page 699.

Fichiers de régulation côté serveur pour autoriser l'accès aux données

Un document Flash peut charger les données depuis une source externe à l'aide de l'un des appels de chargement de données suivants : `XML.load()`, `XML.sendAndLoad()`, `LoadVars.load()`, `LoadVars.sendAndLoad()`, `loadVariables()`, `loadVariablesNum()`, `MovieClip.loadVariables()`, `XMLSocket.connect()` et Flash Remoting (`NetServices.createGatewayConnection`). De plus, un fichier SWF peut importer des bibliothèques partagées à l'exécution (RSL) ou des actifs définis dans un autre fichier SWF, au moment de l'exécution. Par défaut, les données ou le support RSL doivent se trouver dans le même domaine que le fichier SWF qui charge ces données externes ou ce support.

Pour que les fichiers SWF situés dans différents domaines puissent accéder aux données et aux actifs contenus dans des bibliothèques partagées à l'exécution, utilisez un *fichier de régulation inter-domaines*. Il s'agit d'un fichier XML qui permet au serveur d'indiquer que ses données et ses documents sont disponibles pour les fichiers SWF servis par certains domaines ou par tous les domaines. Tout fichier SWF servi par un domaine spécifié par le fichier de régulation du serveur peut accéder aux données, aux ressources et aux RSL de ce serveur.

Si vous chargez des données externes, il est conseillé de créer des fichiers de régulation même si vous n'envisagez pas d'exporter vos fichiers vers Flash Player 7. Si vous utilisez des bibliothèques partagées à l'exécution (RSL), créez des fichiers de régulation si le fichier appelé ou le fichier appelant est publié pour Flash Player 7.

Pour plus d'informations, voir les sections suivantes :

- « Autorisation de chargement de données inter-domaines », à la page 706
- « Emplacements des fichiers de régulation personnalisés », à la page 708
- « A propos des fichiers de régulation XMLSocket », à la page 709

Autorisation de chargement de données inter-domaines

Lorsqu'un document Flash tente d'accéder aux données depuis un autre domaine, Flash Player essaie automatiquement de charger un fichier de régulation depuis ce domaine. Si le domaine du document Flash qui tente d'accéder aux données est inclus dans le fichier de régulation, les données sont automatiquement accessibles.

Les fichiers de régulation doivent être nommés crossdomain.xml et peuvent résider dans le répertoire racine ou dans un autre emplacement du serveur de données avec du code ActionScript supplémentaire (consultez la section « Emplacements des fichiers de régulation personnalisés », à la page 708). Les fichiers de régulation ne fonctionnent que sur des serveurs communiquant en HTTP, HTTPS ou FTP. Le fichier de régulation est spécifique au port et au protocole du serveur dans lequel il réside.

Par exemple, un fichier de régulation situé dans https://www.adobe.com:8080/crossdomain.xml ne s'applique qu'aux appels de chargement de données passés vers www.adobe.com sur HTTPS au port 8080.

Cette règle a une exception : lorsque vous utilisez un objet XMLSocket pour vous connecter à un serveur socket dans un autre domaine. Dans ce cas, un serveur HTTP exécuté sur le port 80 du même domaine que le serveur socket doit fournir le fichier de régulation pour l'appel de la méthode.

Un fichier de régulation XML contient une seule balise `<cross-domain-policy>`, qui contient elle-même aucune ou plusieurs balises `<allow-access-from>`. Chaque balise `<allow-access-from>` contient un attribut, `domain`, qui spécifie une adresse IP exacte, un domaine exact ou un domaine générique (quelconque). Les domaines génériques sont signalés par un astérisque seul * (qui correspond à tous les domaines et à toutes les adresses IP) ou par un astérisque suivi d'un suffixe, qui correspond uniquement aux domaines se terminant par le suffixe spécifié. Les suffixes doivent commencer par un point. Cependant, les domaines génériques suivis de suffixes peuvent correspondre à des domaines qui sont composés uniquement du suffixe sans le point de séparation. Par exemple, adobe.com est considéré comme un élément de *.adobe.com. Les caractères génériques ne sont pas autorisés dans les spécifications de domaine IP.

Si vous spécifiez une adresse IP, seuls les fichiers SWF chargés depuis cette adresse IP à l'aide de la syntaxe IP (par exemple, http://65.57.83.12/flashmovie.swf) sont accessibles ; les fichiers chargés à l'aide d'une syntaxe domaine-nom ne sont pas accessibles. Flash Player n'effectue pas de résolution DNS.

Voici un exemple de fichier de régulation permettant d'accéder à des documents Flash qui proviennent de adobe.com, www.helpexamples.com, *.adobe.com et 105.216.0.40, depuis un document Flash situé sur adobe.com :

```
<?xml version="1.0"?>
<!-- http://www.adobe.com/crossdomain.xml -->
<cross-domain-policy>
  <allow-access-from domain="www.helpexamples.com" />
  <allow-access-from domain="*.adobe.com" />
  <allow-access-from domain="105.216.0.40" />
</cross-domain-policy>
```

Vous pouvez également autoriser l'accès à des documents provenant de tout autre domaine, comme indiqué dans l'exemple suivant :

```
<?xml version="1.0"?>
<!-- http://www.adobe.com/crossdomain.xml -->
<cross-domain-policy>
  <allow-access-from domain="*" />
</cross-domain-policy>
```

Chaque balise `<allow-access-from>` comporte également l'attribut facultatif `secure`. L'attribut `secure` reçoit par défaut la valeur `true`. Vous pouvez définir cet attribut sur `false` si le fichier de régulation figure sur un serveur HTTPS et si vous autorisez les fichiers situés sur un serveur HTTP à charger des données à partir d'un serveur HTTPS.

La définition de l'attribut `secure` sur `false` risque de compromettre la sécurité fournie par le protocole HTTPS.

Si le fichier SWF que vous téléchargez provient d'un serveur HTTPS, alors que le fichier SWF qui le télécharge figure sur un serveur HTTP, vous devez ajouter l'attribut `secure="false"` à la balise `<allow-access-from>`, comme indiqué dans l'exemple de code suivant :

```
<allow-access-from domain="www.adobe.com" secure="false" />
```

Un fichier de régulation ne contenant aucune balise `<autoriser-accès-depuis>` revient à ne pas avoir de régulation sur un serveur.

Emplacements des fichiers de régulation personnalisés

Flash Player 7 (7.0.19.0) prend en charge une méthode appelée `System.security.loadPolicyFile`. Cette méthode permet de spécifier un emplacement personnalisé sur un serveur où réside un fichier de régulation de plusieurs domaines, ce qui implique qu'il ne doit pas nécessairement être dans le répertoire racine. Flash Player 7 (7.0.14.0) recherchait uniquement les fichiers de régulation dans la racine d'un serveur, alors qu'il peut être difficile pour un administrateur de placer des fichiers dans ce type de répertoire. Pour plus d'informations sur la méthode `loadPolicyFile` et les connexions XMLSocket, consultez les sections « A propos des fichiers de régulation XMLSocket », à la page 709 et `loadPolicyFile (méthode security.loadPolicyFile)` du *Guide de référence du langage ActionScript 2.0*.

Si vous utilisez la méthode `loadPolicyFile`, un administrateur de site peut placer le fichier de régulation dans n'importe quel répertoire, à condition que les fichiers SWF qui doivent l'utiliser appellent `loadPolicyFile` pour indiquer à Flash Player l'emplacement du fichier de régulation. Cependant, les fichiers de régulation qui ne sont pas placés dans le répertoire racine ont une étendue limitée. Le fichier de régulation permet uniquement d'accéder aux emplacements situés à son niveau ou à un niveau inférieur dans la hiérarchie du serveur.

La méthode `loadPolicyFile` est disponible uniquement à partir de Flash Player 7 (7.0.19.0) ou ultérieur. Les développeurs de fichiers SWF utilisant la méthode `loadPolicyFile` doivent effectuer l'une des opérations suivantes :

- Obtenir Flash Player 7 (7.0.19.0) ou plus récent.
- Placer un fichier de régulation à l'emplacement par défaut (répertoire racine) ainsi que dans un autre emplacement sur le site d'origine des données. Les versions précédentes de Flash Player utilisent l'emplacement par défaut.

Sinon, les auteurs doivent créer des fichiers SWF pour contrer les problèmes de chargement interdomaines.

ATTENTION

Si votre fichier SWF s'appuie sur `loadPolicyFile`, les visiteurs utilisant une version de Flash Player antérieure à la version 6 ou postérieure à Flash Player 7 (7.0.19.0) n'ont pas de problème. Cependant, les visiteurs équipés de Flash Player 7 (7.0.14.0) ne prendront pas en charge `loadPolicyFile`.

Si vous souhaitez placer un fichier de régulation dans un emplacement personnalisé, vous devez appeler `System.security.loadPolicyFile` *avant* d'émettre des requêtes dépendant de ce type de fichier, comme dans l'exemple suivant :

```
System.security.loadPolicyFile("http://www.adobe.com/folder1/folder2/
  crossdomain.xml");
var my_xml:XML = new XML();
my_xml.load("http://www.adobe.com/folder1/folder2/myData.xml");
```

Vous pouvez charger plusieurs fichiers de régulation avec des étendues qui se chevauchent en utilisant `loadPolicyFile`. Pour toutes les requêtes, Flash Player tente de consulter tous les fichiers dont l'étendue inclut l'emplacement de la requête. Si un fichier de régulation n'accorde pas l'accès interdomaines, cela ne signifie pas que les autres fichiers ne peuvent pas accorder l'accès aux données. Si toutes les tentatives d'accès échouent, Flash Player effectue une recherche dans l'emplacement par défaut du fichier crossdomain.xml (dans le répertoire racine). La requête échoue si aucun fichier de régulation ne figure à l'emplacement par défaut.

A propos des fichiers de régulation XMLSocket

Dans le cas d'une connexion XMLSocket, Flash Player 7 (7.0.14.0) recherchait l'emplacement /crossdomain.xml sur le port 80 d'un serveur HTTP, dans le sous-domaine auquel la connexion était envoyée. Flash Player 7 (7.0.14.0), ainsi que les versions antérieures, limitait les connexions XMLSocket aux ports 1024 et supérieurs. Cependant, à partir de Flash Player 7 (7.0.19.0), le code ActionScript peut informer Flash Player d'un emplacement différent de celui par défaut pour un fichier de régulation utilisant `System.security.loadPolicyFile`. Tout emplacement personnalisé des fichiers de régulation XMLSocket doit demeurer sur un serveur XMLSocket.

Dans l'exemple suivant, Flash Player extrait un fichier de régulation à partir de l'URL spécifiée :

```
System.security.loadPolicyFile("http://www.adobe.com/folder/policy.xml");
```

Les autorisations accordées par ce fichier s'appliquent à l'ensemble du contenu, au même niveau ou à un niveau inférieur dans la hiérarchie du serveur. Par conséquent, si vous tentez de charger les données suivantes, vous vous apercevrez que vous pouvez uniquement charger des données à partir de certains emplacements :

```
myLoadVars.load("http://www.adobe.com/folder/vars.txt"); // autorisé
myLoadVars.load("http://www.adobe.com/folder/dir/vars2.txt"); // autorisé
myLoadVars.load("http://www.adobe.com/elsewhere/vars3.txt"); // non
  autorisé
```

Pour contourner ce problème, vous pouvez charger plusieurs fichiers de régulation dans un seul fichier SWF avec `loadPolicyFile`. Flash Player attend toujours la fin des téléchargements du fichier de régulation avant de refuser une requête nécessitant ce type de fichier. Flash Player consulte l'emplacement par défaut de crossdomain.xml si aucune régulation n'a été autorisée dans le fichier SWF.

Il est possible de récupérer des fichiers de régulation directement depuis un serveur XMLSocket grâce à une syntaxe particulière:

```
System.security.loadPolicyFile("xmlsocket://adobe.com:414");
```

Dans cet exemple, Flash Player peut récupérer un fichier de régulation au niveau du port et de l'hôte spécifiés. Tout port peut être utilisé si le fichier de régulation ne figure pas dans le répertoire par défaut (racine). Sinon, le port est limité à 1024 ou toute valeur supérieure (comme pour les versions précédentes des lecteurs). Lorsque la connexion au port spécifié est établie, Flash Player transmet la chaîne `<policy-file-request />`, suivie d'un octet nul.

Le serveur XMLSocket peut être configuré pour servir les fichiers de régulation de la façon suivante :

- Pour servir les fichiers de régulation et les connexions normales de socket sur le même port. Le serveur doit attendre la réception de la chaîne `<policy-file-request />` avant de transmettre un fichier de régulation.
- Pour distribuer les fichiers de régulation et les connexions normales via des ports différents. Dans ce cas, le serveur peut transmettre un fichier dès qu'une connexion est établie au niveau du port dédié aux fichiers de régulation.

Le serveur doit envoyer un octet null pour terminer un fichier de régulation avant de clore la connexion. Si le serveur ne termine pas la connexion, Flash Player s'en charge lors de la réception de l'octet null de terminaison.

La syntaxe des fichiers de régulation transmis par un serveur XML Socket est identique à celle des autres fichiers de régulation, à l'exception que ces fichiers doivent également spécifier les ports auxquels ils permettent d'accéder. Les ports accessibles sont spécifiés par l'attribut `to-ports` dans la balise `<allow-access-from>`. Si un fichier de régulation est inférieur au port 1024, il peut accorder l'accès à n'importe quel port. S'il provient du port 1024 ou plus élevé, il peut autoriser l'accès aux ports à compter de 1024. Les numéros de port uniques, les séries de ports et les caractères génériques sont autorisés. Le code suivant est un exemple de fichier de régulation XMLSocket :

```
<cross-domain-policy>
<allow-access-from domain="*" to-ports="507" />
<allow-access-from domain="*.adobe.com" to-ports="507,516" />
<allow-access-from domain="*.helpexamples.com" to-ports="516-523" />
<allow-access-from domain="www.adobe.com" to-ports="507,516-523" />
<allow-access-from domain="www.helpexamples.com" to-ports="*" />
</cross-domain-policy>
```

La possibilité de se connecter aux ports dont le numéro est inférieur à 1024 étant une nouveauté de Flash Player 7 (7.0.19.0) et ultérieur, les fichiers de régulation chargés avec `loadPolicyFile` doivent spécifier cette possibilité, même lors de la connexion d'un fichier SWF à son propre sous-domaine.

Accès du protocole HTTP vers le protocole HTTPS entre les fichiers SWF

Pour qu'un fichier SWF d'un domaine soit accessible à un fichier SWF d'un autre domaine, vous devez utiliser un gestionnaire ou une méthode `allowDomain`. Toutefois, si le fichier SWF accédé est hébergé sur un site qui utilise un protocole sécurisé (HTTPS), le gestionnaire ou la méthode `allowDomain` interdit l'accès à tout SWF hébergé sur un site utilisant un protocole non sécurisé. Pour autoriser un tel accès, vous devez utiliser les instructions `LocalConnection.allowInsecure Domain()` ou `System.security.allowInsecureDomain()`. Pour plus d'informations, consultez la section « Autorisation d'accès du protocole HTTP au protocole HTTPS entre fichiers SWF », à la page 712.

Autorisation d'accès du protocole HTTP au protocole HTTPS entre fichiers SWF

En plus des règles de correspondance exacte de domaines, vous devez autoriser de façon explicite l'accès aux fichiers situés sur des sites qui utilisent un protocole sécurisé (HTTPS) par des fichiers hébergés sur des sites utilisant des protocoles non sécurisés. Selon que votre fichier est publié pour Flash Player 6 ou ses versions ultérieures, vous devez implémenter l'une des deux instructions `allowDomain` (consultez la section « Accès inter-domaines et accès aux sous-domaines entre fichiers SWF », à la page 699) ou utiliser les nouvelles instructions `LocalConnection.allowInsecure Domain` ou `System.security.allowInsecureDomain()`.

Par exemple, si un fichier SWF à l'adresse http://www.adobe.com doit pouvoir accéder au fichier SWF à l'adresse https://www.adobe.com/data.swf, ajoutez le code suivant au fichier data.swf :

```
// Dans data.swf
System.security.allowInsecureDomain("www.adobe.com");
my_lc.allowInsecureDomain = function(sendingDomain:String):Boolean {
  return (sendingDomain == "www.adobe.com");
};
```

AVERTISSEMENT

L'implémentation d'une instruction `allowInsecureDomain()` compromet la sécurité assurée par le protocole HTTPS. Vous devez effectuer ces modifications uniquement si vous ne pouvez pas réorganiser votre site de sorte que tous les fichiers SWF soient servis par le protocole HTTPS.

Le code suivant présente un exemple des modifications qu'il peut être nécessaire d'apporter :

```
// Commandes dans un fichier SWF Flash Player 6 situé sur
// https://www.adobe.com
// pour autoriser l'accès par des fichiers SWF Flash Player 7 hébergés
// sur http://www.adobe.com ou sur http://www.helpexamples.com
System.security.allowDomain("helpexamples.com");
ma_lc.allowDomain = function(sendingDomain) {
  return(sendingDomain=="helpexamples.com");
}
// Commandes correspondantes dans un fichier SWF Flash Player 7
// pour autoriser l'accès par des fichiers SWF Flash Player 7 hébergés
// sur http://www.adobe.com ou sur http://www.helpexamples.com
System.security.allowInsecureDomain("www.adobe.com",
  "www.helpexamples.com");
my_lc.allowInsecureDomain = function(sendingDomain) {
  return(sendingDomain=="www.adobe.com" ||
    sendingDomain=="www.helpexamples.com");
}
```

Il peut également être nécessaire d'ajouter des instructions similaires dans vos fichiers si vous ne les utilisez pas actuellement. Une modification peut s'avérer nécessaire même si les deux fichiers résident dans le même domaine (par exemple, un fichier situé sur http://www.domain.com appelle un fichier situé sur https://www.adobe.com).

Pour résumer, il peut être nécessaire de modifier vos fichiers pour ajouter ou modifier des instructions si vous publiez des fichiers pour Flash Player 7 ou une version ultérieure qui correspondent aux critères suivants :

- Vous avez implémenté des scripts entre des fichiers SWF (à l'aide de `loadMovie()`, `MovieClip.loadMovie()`, `MovieClipLoader.LoadClip()` ou d'objets Local Connection).
- Le fichier appelant n'est pas hébergé à l'aide d'un protocole HTTPS et le fichier appelé est sécurisé.

Vous devez effectuer les modifications suivantes :

- Si le fichier appelé est publié pour Flash Player 7, incluez-y `System.security.allowInsecureDomain` ou `LocalConnection.allowInsecureDomain` en utilisant la correspondance exacte de domaine, comme dans les exemples de code précédents de cette section.
- Si le fichier appelé est publié pour Flash Player 6 ou une version antérieure, et que le fichier appelant et le fichier appelé résident dans le même domaine (par exemple, si un fichier sur http://www.adobe.com appelle un fichier situé sur https://www.adobe.com), aucune modification n'est nécessaire.
- Si le fichier appelé est publié pour Flash Player 6, que les fichiers ne se trouvent pas dans le même domaine et que vous ne souhaitez pas exporter le fichier appelé vers Flash Player 7, modifiez le fichier appelé en ajoutant ou en modifiant une instruction `System.security.allowDomain` ou `LocalConnection.allowDomain` à l'aide de la correspondance exacte de domaine, comme dans les exemples de code précédents de cette section.
- Si le fichier appelé est publié pour Flash Player 6 et que vous souhaitez l'exporter vers Flash Player 7, incluez-y `System.security.allowInsecureDomain` ou `LocalConnection.allowInsecureDomain` en utilisant la correspondance exacte de domaine, comme dans les exemples de code précédents de cette section

- Si le fichier appelé est publié pour Flash Player 5 ou une version antérieure et que les deux fichiers ne se trouvent pas dans le même domaine, vous pouvez procéder de l'une des deux manières suivantes. Vous pouvez soit porter le fichier appelé sur Flash Player 6 et ajouter ou modifier une instruction `System.security.allowDomain` en utilisant la correspondance de nom de domaine exacte, soit porter le fichier appelé sur Flash Player 7 et inclure une instruction `System.security.allowInsecureDomain` dans le fichier appelé en utilisant le filtrage de domaine, comme illustré dans les exemples de code, plus haut dans cette section.

CHAPITRE 17

Recommandations et conventions de programmation pour ActionScript 2.0

17

Les concepteurs et les développeurs Adobe Flash doivent rédiger leur code et structurer leurs applications selon une logique intuitive et profitable, non seulement pour eux-mêmes, mais également pour les autres personnes travaillant sur le même projet. Ce point est particulièrement important dans le cas de fichiers FLA volumineux ou faisant appel à de nombreuses ressources. Le respect des recommandations et des conventions de programmation permet aux membres de votre équipe de comprendre la structure des fichiers et du code ActionScript, donc de travailler efficacement. Ce document assiste dans le dévelopement Flash et le processus de programmation.

Il n'est pas rare que plusieurs concepteurs ou développeurs travaillent sur le même projet Flash. Il est alors dans l'intérêt de tous d'appliquer les recommandations standard d'utilisation de Flash, d'organisation des fichiers et de rédaction du code ActionScript 2.0. Les sections de ce chapitre présentent les meilleures méthodes d'écriture de code ActionScript et certaines sections de *Utilisation de Flash* couvrent les meilleures méthodes sur l'emploi de l'outil de programmation Flash.

Les recommandations suivantes insistent sur l'importance de l'uniformité du code, notamment pour les personnes qui apprennent à utiliser Flash et à programmer en langage ActionScript. Vous devez appliquer ces pratiques de façon rigoureuse, que vous soyez concepteur ou développeur, seul ou membre d'une équipe.

- Travailler sur des documents Flash ou ActionScript

  En adoptant des pratiques cohérentes et efficaces, vous accélérerez votre rendement de travail. Il est plus rapide d'appliquer des conventions de programmation établies et plus facile de comprendre et de mémoriser la structure de vos documents lorsque vous devez les modifier ultérieurement. En outre, votre code devient généralement plus portable dans le cadre d'un projet plus important, et plus facile à réutiliser.
- Partager des fichiers FLA ou AS

  Toute personne intervenant sur le document peut trouver et comprendre rapidement ActionScript, modifier le code de façon cohérente et rechercher et modifier des ressources.

- Travailler sur ces applications

  Plusieurs auteurs peuvent travailler sur une application de façon plus efficace, tout en réduisant les conflits. Les administrateurs de projet ou de site peuvent gérer et structurer des projets ou des applications complexes recelant moins de conflits ou de redondances.
- Apprendre ou enseigner le travail sur des documents Flash ou ActionScript

  L'application des recommandations et des conventions de programmation lors de la création d'applications évite les apprentissages répétés des méthodologies particulières. Les personnes en cours de formation sur Flash bénéficient de l'application systématique des meilleures pratiques, dans la mesure où elles simplifient la structuration du code et permettent d'apprendre le langage plus rapidement et plus en profondeur.

Des techniques cohérentes et l'application des conseils suivants facilitent la tâche des personnes en formation à Flash ou travaillant dans des équipes. Une méthodologie rigoureuse permet de mieux mémoriser la structure d'un document lorsque vous travaillez en solo, notamment si vous n'avez pas manipulé les fichiers FLA depuis un certain temps.

Nous pourrions citer bien d'autres raisons d'apprendre et mettre en œuvre les meilleures pratiques. Vous les découvrirez lors de la lecture de ce chapitre et les appliquerez rapidement sans plus y penser. Tenez compte des sections suivantes lorsque vous travaillez sous Flash ; vous pouvez appliquer ces recommandations en tout ou partie. Vous pouvez également les modifier pour les adapter à votre méthode de travail. Vous trouverez dans ce chapitre de nombreux conseils qui permettent d'utiliser Flash et d'écrire du code ActionScript de façon rigoureuse.

Ce chapitre aborde les sujets suivants sur les conventions de code et les meilleures méthodes :

Conventions d'appellation

De manière générale, 80 % du temps de développement est consacré au débogage, à la résolution des dysfonctionnements et aux opérations de maintenance générale, notamment pour les gros projets. Les petits projets impliquent également une période d'analyse et de correction du code. La lisibilité du code est importante tant pour vous que pour les autres membres de l'équipe. En respectant les conventions d'appellation, vous améliorez la lisibilité, donc le flux de travail, et êtes mieux à même de déceler et corriger les erreurs de code éventuelles. Tous les programmeurs utilisent une méthode normalisée de rédaction du code. Leurs projets en sont optimisés de bien des façons.

L'utilisation de conventions d'appellation pour les noms des variables remplit les objectifs essentiels suivants :

- Meilleure lisibilité du code permettant d'identifier immédiatement le type de données d'une variable. Le travail des stagiaires qui apprennent la programmation ou des développeurs qui découvrent votre code s'en trouve simplifié.
- Elles sont faciles à rechercher et à remplacer si besoin.
- Réduction des conflits liés aux mots réservés et aux éléments de langage.
- Elles peuvent vous aider à distinguer des variables à différents niveaux (variables locales, propriétés de classes, paramètres, etc.).

Les sections suivantes contiennent des instructions d'appellation pour écrire du code ActionScript, comme appellation de fichiers, de variables, de constantes, de composants, etc. « Mise en forme de la syntaxe ActionScript », à la page 750 présente les conventions de mise en forme spécifiques à ActionScript et courantes dans d'autres langages de programmation. « Conventions de programmation ActionScript », à la page 731 présente les conventions de code spécifiques à l'écriture d'ActionScript et au développement avec Flash 8.

REMARQUE

Flash Player 7 et ses versions ultérieures respectent rigoureusement la spécification de langage ECMAScript (ECMA-262) version 3. Il est recommandé de consulter cette norme pour obtenir plus de détails sur le fonctionnement du langage. (Consultez le site www.ecma-international.org/publications/standards/Ecma-262.htm).

Cette section inclue les rubriques suivantes :

Instructions générales d'appellation

Cette section couvre les instructions d'appellation relatives à la rédaction du code ActionScript. Les conventions d'appellation sont importantes pour la rédaction de code logique. Le but principal est d'améliorer la lisibilité de votre code ActionScript 2.0. N'oubliez pas que les variables doivent porter un nom unique. Dans Flash Player 7 et les versions ultérieures, les noms respectent la casse. N'utilisez pas le même nom en changeant simplement la casse, car les programmeurs risquent d'être induits en erreur à la lecture de votre code et cela peut occasionner des problèmes des versions antérieures de Flash ne tenant pas compte de la casse. Gardez ces instructions à l'esprit lorsque vous nommez des éléments comme les variables, les fichiers et les classes dans Flash :

- Limitez l'utilisation d'abréviations.

  Utilisez les abréviations de manière cohérente Chaque abréviation doit être associée à une chose précise. Par exemple, l'abréviation « sec » peut aussi bien représenter « section » que « seconde ».
- Concaténez des mots pour former des noms.

  Mélangez des majuscules et des minuscules lorsque vous concaténez des mots, de façon à les distinguer les uns des autres pour une meilleure lisibilité. Par exemple, sélectionnez `monToucan` et non pas`montoucan`.
- Nommez un fichier en décrivant le processus ou l'article, tel que `ajoutUtilisateur`.
- Evitez d'utiliser des noms non descriptifs pour les méthodes ou les variables.

  Par exemple, si vous extrayez des données correspondant au nom d'utilisateur, utilisez la méthode `getUserName()` et non pas `getData()` qui est moins explicite. Cet exemple indique ce qui doit se passer et non pas comment y parvenir.
- Gardez tous des noms aussi courts que possible.

  Souvenez-vous de garder des noms descriptifs.

Les sections suivantes décrivent plus en détails l'appellation des éléments comme les variables, les classes, les packages et les constantes dans votre code.

Utilisation restreinte des mots réservés et des éléments de langage

Lorsque vous choisissez les noms des occurrences et des variables, évitez les mots réservés, car ils provoqueraient des erreurs dans votre code. Les mots réservés incluent les *mots-clés* du langage ActionScript.

De même, dans le langage ActionScript 2.0, n'utilisez pas de termes (appelés *éléments de langage*) comme noms d'occurrence ou de variable. Les éléments ActionScript comprennent les noms de classe, les noms de classe de composants, les noms de méthodes et de propriétés et les noms d'interface.

AVERTISSEMENT	Ne tentez pas d'éviter les conflits liés aux mots réservés par l'emploi de majuscules/minuscules. Par exemple, nommer une occurrence de la classe TextField `textfield` (ce qui n'entraîne pas de conflit avec TextField puisque Flash respecte la casse) n'est vraiment pas recommandé.

Le tableau suivant regroupe les mots-clés réservés du langage ActionScript 2.0 qui provoquent des erreurs dans vos scripts lorsqu'ils sont utilisés comme noms de variable :

add	and	break	case
catch	class	continue	default
delete	do	dynamic	else
eq	extends	false	finally
for	function	ge	get
gt	if	ifFrameLoaded	implements
import	in	instanceof	interface
intrinsic	le	it	ne
new	not	null	on
onClipEvent	or	private	public
return	set	static	super
switch	tellTarget	this	throw
try	typeof	non défini	var
void	while	with	

Les mots suivants sont réservés pour les prochaines versions de Flash, sur la base de la spécification ECMAScript (ECMA-262) version 4. Evitez d'utiliser ces mots dans la mesure où ils pourraient être utilisés dans des futures versions de Flash.

as	abstract	Boolean	bytes
char	const	debugger	double
enum	export	final	float
goto	is	long	namespace
native	package	protected	short
synchronized	throws	transient	use
volatile			

Appellation des variables

Les noms de variables doivent uniquement contenir des lettres, des numéros et des signes dollar (`$`). Ne faites pas commencer les noms de variable par un numéro. Les variables doivent être uniques et respectent la casse sous Flash Player 7 et les versions ultérieures. Par exemple, évitez les noms de variable suivants :

```
my/warthog = true;   // Inclut une barre oblique
my warthogs = true;  // inclut un espace
my.warthogs = true; // Inclut un point
5warthogs = 55;    // commence par un numéro
```

Appliquez un typage strict des données à vos variables dans la mesure du possible, car ceci facilite les opérations suivantes :

- Ajoute de la fonctionnalité de saisie automatique pour accélérer le codage.
- Génère des erreurs dans le panneau de sortie de manière à ce que vous ne rencontriez pas d'échec silencieux lorsque vous compilez votre fichier SWF. Ces erreurs vous permettent de déterminer et de résoudre les problèmes dans vos applications.

Pour ajouter un type de données à vos variables, vous devez définir la variable avec le mot-clé `var`. Dans l'exemple suivant, lors de la création de l'objet LoadVars, appliquez un typage strict des données :

```
var paramsLv:LoadVars = new LoadVars();
```

La saisie stricte des données permet de bénéficier de la fonction de saisie automatique et de s'assurer que la valeur de `paramsLv` contient un objet LoadVars. Il permet également de s'assurer que l'objet LoadVars n'est pas utilisé pour stocker des données de type numérique ou chaîne. Dans la mesure où la saisie strict des données repose sur le mot-clé `var`, vous ne pouvez pas l'ajouter aux variables ou paramètres globaux au sein d'un objet ou d'un tableau. Pour plus d'informations sur la saisie stricte des variables, consultez la section « Affectation des types de données et typage strict », à la page 45.

REMARQUE	Le typage strict des données ne ralentit pas le fichier SWF. Le type est vérifié lors de la compilation (lorsque le fichier SWF est créé) et non pas pendant son exécution.

Respectez les consignes suivantes lorsque vous nommez des variables dans votre code :

- Toutes les variables doivent porter un nom unique.
- N'employez pas le même nom de variable avec des casses différentes.

  Par exemple, d'employez pas `prénom` et `préNom` en tant que variables différentes dans votre application. Bien que les noms tiennent compte des majuscules et des minuscules dans Flash Player 7 et les versions ultérieures, n'utilisez pas le même nom en changeant simplement la casse car les programmeurs risquent de les confondre et des problèmes de compatibilité risquent de survenir dans des versions précédentes de Flash qui ne tiennent pas compte de la casse.
- N'utilisez pas des mots qui font partie du langage ActionScript 1.0 ou 2.0 en tant que noms de variable.

  Plus précisément, n'utilisez jamais des mots clés en tant que noms d'occurrence, car ils génèrent des erreurs dans votre code. Ne comptez pas sur une différence de casse pour éviter les conflits et faire fonctionner votre code.
- N'employez pas des variables qui font partie d'éléments de programmation courante.

  N'utilisez pas d'éléments de langage lorsque vous savez qu'ils appartiennent à d'autres langages de programmation, même si Flash ne les utilise pas ou ne les prend pas en charge en tant que tel. Par exemple, n'utilisez pas les mots clés suivants en tant que variables :

```
textfield = "myTextField";
switch = true;
new = "funk";
```

- Ajoutez toujours des annotations de type de données à votre code.

  L'ajout d'annotations à vos variables concernant le type, appelées également « utilisation de types de données strictes avec vos variables » ou « typage renforcé de vos variables », est important et obtient les effets suivants :

  - Génère des erreurs au moment de la compilation de sorte que votre application ne tombera pas en panne silencieusement à l'exécution.
  - Fournit un code d'achèvement.
  - Permet aux utilisateurs de comprendre votre code.

  Pour plus d'informations sur l'ajout d'annotations de type, consultez « Affectation des types de données et typage strict », à la page 45.
- N'abusez pas du type objet.

  Les annotations de type de données doivent être précises pour améliorer les performances. N'utilisez un type d'objet que lorsqu'il n'y a pas d'autre solution raisonnable.
- Essayez de réduire les noms de variable au minimum tout en en préservant la clarté.

  Choisissez des noms de variables descriptifs, mais pas trop longs, ni trop complexes.
- N'employez que des noms de variable à un caractère pour optimiser les boucles.

  Certains développeurs ont recours à des variables d'un seul caractère pour les variables temporaires dans les boucles (telles que `i`, `j`, `k`, `m` et `n`). Réservez exclusivement les noms de variable d'une lettre aux index de boucle courts. Vous pouvez également les utiliser lorsque l'optimisation et la vitesse sont essentielles. L'exemple suivant illustre ces opérations :

  ```
  var fontArr:Array = TextField.getFontList();
  fontArr.sort();
  var i:Number;
  for (i = 0; i<fontArr.length; i++) {
    trace(fontArr[i]);
  }
  ```
- Les noms de variables commencent par une minuscule.

  Les noms commençant par une majuscule sont réservés aux classes, aux interfaces, etc.
- Employez des majuscules et des minuscules pour les mots concaténés.

  Par exemple, utilisez `maPolice` au lieu de `mapolice`.
- N'utilisez pas des acronymes et des abréviations.

  Une seule exception à cette règle : lorsque les acronymes ou les abréviations constituent le moyen standard d'exprimer le terme (tel que HTML ou CFM). Dans le cas d'acronymes courants, combinez la casse. Par exemple, utilisez `newHtmlParser` plutôt que `newHTMLParser`, pour plus de lisibilité.
- Utilisez des paires complémentaires lorsque vous créez des ensembles apparentés de noms de variable.

Par exemple, vous pouvez utiliser des paires complémentaires pour indiquer les scores minimum et maximum d'un jeu :

```
var minScoreNum:Number = 10; //score minimum
var maxScoreNum:Number = 500; //score maximum
```

Appellation des constantes

Vous pouvez utiliser des constantes pour les situations où vous devez faire référence à une propriété dont la valeur ne change jamais. Ceci permet d'identifier les erreurs typographiques qui risquent de passer inaperçues avec des littéraux. Ceci vous permet également de centraliser la modification des valeurs.

Les variables doivent être en minuscules ou comprendre un mélange de minuscules et majuscules. Toutefois, utilisez les instructions suivantes pour les noms des constantes statiques (variables qui ne changent pas) :

- Les constantes doivent être en majuscules.
- Les mots distincts doivent être séparés par des traits de soulignement.

L'exemple de code ActionScript suivant permet de retrouver ces consignes :

```
var BASE_URL:String = "http://www.adobe.com"; // constante
var MAX_WIDTH:Number = 10;  // constante
```

Ne codez pas directement les constantes numériques, sauf si leur valeur est 1, 0 ou -1, que vous pouvez utiliser dans le cadre d'une boucle `for` en tant que valeur de compteur.

Appellation des variables booléennes

Commencez les variables booléennes par le mot « est » (parce que, de par sa nature, une valeur booléenne « est » ou « n'est pas »). De ce fait, utilisez le code suivant lorsqu'un bébé est ou n'est pas une fille (valeur booléenne) :

```
isGirl
```

Ou pour une variable indiquant si un utilisateur est connecté (ou non), vous pouvez utilisez ce qui suit :

```
isLoggedIn
```

Appellation des fonctions et des méthodes

Respectez les consignes suivantes lorsque vous nommez des fonctions et des méthodes dans votre code. Pour plus d'informations sur l'écriture des fonctions et des méthodes, consultez le Chapitre 5, « Fonctions et méthodes ».

- Choisissez des noms descriptifs.
- Employez des majuscules et des minuscules pour les mots concaténés.

  `chanterFort()` est un bon exemple.
- Faites commencer les noms des fonctions et des méthodes par une minuscule.
- Décrivez la valeur renvoyée lorsque vous créez les noms de fonction.

  Par exemple, si vous renvoyez le nom d'une chanson, nommez la fonction `saisirChansonCourante()`.
- Etablissez une norme d'appellation pour les fonctions similaires.

  ActionScript 2.0 n'autorise pas les surcharges Dans le cadre de la programmation orientée objet, la*surcharge* consiste à autoriser des comportements différents pour vos fonctions, selon les types de données qui leurs sont transmis.
- Nommez les méthodes comme des verbes.

  Vous pouvez concaténer le nom, mais il doit contenir un verbe. Les verbes sont conseillés car les méthodes exécutent une action sur un objet.

Voici quelques exemples de noms de méthodes :

```
sing();
boogie();
singLoud();
danceFast();
```

Appellation des classes et des objets

Respectez les consignes suivantes pour nommer la classe et le fichier ActionScript lorsque vous créez un nouveau fichier de classe. Consultez les exemples suivants de noms de classe pour un formatage correct :

```
class Widget;
class PlasticWidget;
class StreamingVideo;
```

Vous pouvez disposer de variables de membre publiques et privées dans une classe. La classe peut contenir des variables que les utilisateurs ne doivent par définir ou auxquelles ils ne peuvent pas accéder. Rendez ces variables privées et autorisez les utilisateurs à accéder aux valeurs en n'utilisant que des méthodes de type lecture/définition.

Les recommandations suivantes s'appliquent au choix des noms de classes :

- Faites commencer le nom d'une classe par une majuscule.
- Ecrivez les noms de classe en alternant les majuscules et les minuscules lorsque ces dernières comportent des mots composés ou concaténés.

  Les composants qui utilisent des mots composés ou concaténés commencent par une lettre en majuscule. `NouveauMembre` est un bon exemple.
- Les noms de classe sont généralement des noms ou des noms qualifiés.

  Le qualificateur décrit le nom ou l'expression. Par exemple, au lieu de « membre », vous pouvez qualifier le nom de la façon suivante, `NouveauMembre` ou `AncienMembre`.
- Il est plus important d'utiliser des noms clairs, que courts
- N'utilisez pas des acronymes et des abréviations.

  Une seule exception à cette règle : lorsque les acronymes ou les abréviations constituent le moyen standard d'exprimer le terme (tel que HTML ou CFM). Dans le cas d'acronymes courants, combinez la casse. Par exemple, utilisez `NewHtmlParser` plutôt que `NewHTMLParser`, pour plus de lisibilité.
- Utilisez des noms cohérents et simples qui décrivent le contenu des classes.

  Pour éviter d'être vague ou d'induire en erreur, employez des termes génériques.
- Parfois un nom de classe peut être un mot composé.

  Un qualificateur peut décrire le nom ou l'expression. Par exemple, au lieu de « membre », vous pouvez qualifier le nom de la façon suivante, `NouveauMembre` ou `AncienMembre`.
- Ne mettez pas les mots que vous utilisez dans le nom de classe au pluriel (tel que `Sorcières` ou `Pirates`).

  Dans la plupart des cas, il est préférable de laisser les mots en tant que noms qualifiés. Le qualificateur décrit le nom ou l'expression. Par exemple, au lieu de « chat » ou « boucanier », vous pouvez qualifier le nom de la façon suivante, `ChatNoir` ou `ChatCapricieux`.
- N'utilisez pas de nom de classe dans les propriétés de cette classe en raison des risques de redondance.

  Par exemple, le nom suivant n'a aucun sens `Chat.moustachesChat`. `Chat.moustaches` est préférable.
- N'utilisez pas de noms qui pourraient être pris pour des verbes.

  Par exemple, `Court` ou `Jardine`. L'utilisation de ces noms peut prêter à confusion avec les méthodes, les états ou d'autres applications.
- Utilisez des noms de classe uniques pour chaque classe dans une seule application.

- Nommez les classes de manière à ce qu'elles n'entrent pas en conflit avec les noms des classes incorporées dans Flash.
- Essayez de transmettre la relation qu'une classe possède avec une hiérarchie.

  Cela permet de révéler la relation d'une classe au sein d'une application. Par exemple, vous pouvez disposer d'une interface appelée Gadget. L'implémentation de Gadget pouvant être GadgetPlastique, GadgetAcier et PetitGadget.

Pour plus d'informations sur les interfaces, consultez le Chapitre 8, « Interfaces ».

Appellation des packages

Les noms de package utilisent généralement des conventions d'appellation en « domaine inversé ». Dans le cas du domaine adobe.com, `com.adobe` est un exemple de nom de domaine inversé et `org.votredomaine`, le nom de domaine inversé de votredomaine.org.

Respectez les consignes suivantes lorsque vous nommez les packages.

- Notez le préfixe du nom d'icône-objet en minuscules.

  Par exemple, `com`, `mx` ou `org`.
- Placez les classes apparentées (classes avec les fonctionnalités associées) dans le même package.
- Faites commencer les noms d'icône-objets par un préfixe cohérent.

  Par exemple, vous pouvez utiliser `com.adobe.nomProjet` pour maintenir une cohérence. `com.adobe.docs.apprendreAS2.Utilisateurs` est un autre exemple relatif au *Guide de référence Formation à ActionScript 2.0*.
- Utilisez un nom d'icône-objet clair et explicite.

  Il est important dans la mesure où le nom décrit les responsabilités du package. Par exemple, si votre package s'appelle `Pentagones` et permet de dessiner différents types de pentagones avec l'API de dessin de Flash, son nom serait `com.adobe.docs.as2.Pentagones`.
- Employez des majuscules et des minuscules pour les noms composés ou concaténés de package.

  `nomPackage` est une exemple de nom composé et concaténé pour un package N'oubliez pas d'utiliser des lettres minuscules pour le préfixe (com, org, etc.).
- N'utilisez pas des caractères soulignés ou le symbole dollar.

Appellation des interfaces

Le fait de commencer les noms d'interface par un « I » majuscule permet de les distinguer des classes. Le nom d'interface `IDossiersEmployes` est un mot concaténé qui commence par une majuscule suivie de minuscules comme suit:

```
interface IEmployeeRecords{}
```

Les conventions suivantes s'appliquent également :

- Les noms d'interface commencent par une majuscule.

  C'est la même chose que pour les noms de classe.
- Les noms d'interface sont généralement des adjectifs.

  `Imprimable` est un bon exemple.

Pour plus d'informations sur les interfaces, consultez le Chapitre 8, « Interfaces ».

Appellation des composants personnalisés

Les noms de composant doivent commencer par une majuscule et les mots concaténés doivent être également comporter une majuscule initiale. Par exemple, les éléments d'interface utilisateur par défaut utilisent des mots concaténés et des casses mixtes :

- CheckBox
- ComboBox
- DataGrid
- DateChooser
- DateField
- MenuBar
- NumericStepper
- ProgressBar
- RadioButton
- ScrollPane
- TextArea
- TextInput

Les composants qui n'utilisent pas de mots concaténés commencent par une lettre en majuscule.

Si vous développez des composants personnalisés, appliquez une convention d'appellation pour éviter tout risque de confusion avec les composants d'Adobe. Les noms de vos composants doivent différer du jeu de composants par défaut de Flash. L'application de conventions d'appellation permet d'éviter les conflits entre les noms.

N'oubliez pas que les conventions d'appellation de cette section servent uniquement de directives. Il est essentiel d'appliquer un profil d'appellation qui convienne à votre activité de façon systématique.

Insertion de commentaires dans le code

Cette section décrit l'utilisation des commentaires dans votre code. Les commentaires documentent les décisions prises lors de la rédaction du code, expliquant le *pourquoi* et le *comment*. Par exemple, vous devez décrire toute solution de contournement dans les commentaires. Un autre développeur pourra alors retrouver aisément le code concerné pour le mettre à jour ou le corriger. Enfin, si le problème est corrigé dans une version ultérieure de Flash ou Flash Player, la solution de contournement devient alors inutile.

Pour plus d'informations sur l'écriture de commentaires dans le code ActionScript, consultez les sections suivantes :

- « Rédaction de commentaires appropriés », à la page 728
- « Insertion de commentaires dans les classes », à la page 730

Rédaction de commentaires appropriés

L'ajout de commentaires cohérents dans votre code ActionScript 2.0 permet de décrire les zones de code complexes ou les interactions importantes qui, autrement, demeureraient confuses. Les commentaires doivent expliquer clairement l'objectif du code et ne pas se contenter de le paraphraser. Toute section qui ne s'explique pas d'elle-même dans le code doit faire l'objet de commentaires.

Si vous utilisez l'outil de formatage automatique, vous constaterez que les commentaires en fin de ligne (consultez « Commentaires en fin de ligne », à la page 103) passent sur la ligne suivante. Vous pouvez les ajouter après avoir formaté votre code ou les changer de place après avoir utilisé l'outil de formatage automatique.

Pour plus d'informations sur l'exploitation des commentaires dans les classes, consultez la section « Insertion de commentaires dans les classes », à la page 730.

Respectez les consignes suivantes lorsque vous insérez des commentaires dans votre code :

- Créez des blocs de commentaires (/* et */) pour les commentaires nécessitant plusieurs lignes et des commentaires d'une seule ligne (//) pour les commentaires courts.

  Vous pouvez également utiliser un *commentaire en bout de ligne* sur la même ligne que le code ActionScript si besoin.
- Assurez-vous de ne pas utiliser de commentaires pour traduire votre code ActionScript.

  Il n'est pas nécessaire de commenter sur des éléments qui sont clairs dans le code ActionScript.
- Le commentaire sur les éléments qui ne sont pas clairement évidents dans le code.

  En particulier, ajoutez des commentaires lorsque le sujet n'est pas décrit dans les paragraphes alentours
- N'utilisez pas de commentaires superflux.

  Une ligne de commentaires superflux comprend souvent des signes égal (=) ou des astérisques (*). Dans ce cas, utilisez des espaces blancs pour séparer les commentaires du code ActionScript.

REMARQUE	Si vous utilisez l'outil de format automatique pour formater ActionScript, l'espace blanc est supprimé. N'oubliez pas de le rétablir ou d'utiliser des commentaires d'une ligne (//) pour préserver l'espacement. Ces lignes sont faciles à supprimer après avoir formaté le code.

- Avant de déployer votre projet, supprimez les commentaires superflux du code.

  Si votre code ActionScript comporte trop de commentaires, il peut être nécessaire d'en rédiger de nouveau certaines sections. Le sentiment de devoir ajouter de nombreux commentaires pour décrire le fonctionnement du code ActionScript dénote souvent une logique de programmation médiocre.

REMARQUE	L'emploi de commentaires est particulièrement important lorsque le code ActionScript est destiné à la formation. Par exemple, ajoutez des commentaires à votre code si vous créez des exemples d'application dans le but de former les utilisateurs à Flash ou si vous rédigez votre propre documentation ActionScript.

Insertion de commentaires dans les classes

Les deux types de commentaires dans une classe typique ou un fichier d'interface sont des *commentaires de documentation* et des *commentaires d'implémentation*.

REMARQUE	Les commentaires de documentation et d'implémentation ne sont pas formellement représentés dans le langage ActionScript. Toutefois, ils sont couramment utilisés par les développeurs lors de l'écriture de fichiers de classe et d'interface.

Les commentaires de documentation décrivent le cahier des charges du code, mais pas son implémentation. Les commentaires d'implémentation décrivent le code ou l'implémentation de sections spécifiques du code. Les commentaires de documentation sont séparés par des /** et des */, tandis que les commentaires d'implémentation sont séparés par des /* et des */.

Servez-vous des commentaires de documentation pour décrire les interfaces, les classes, les méthodes et les constructeurs. Incluez un commentaire de documentation par classe, interface ou membre, et placez-le directement avant la déclaration. Si vous devez faire des ajouts à vos commentaires de documentation, utilisez des commentaires d'implémentation (sous forme de blocs de commentaires ou de commentaires sur une ligne).

Commencez vos classes par un commentaire standard, au format suivant :

```
/**
  Classe utilisateur
  version 1.2
  3/21/2004
  copyright Adobe Systems Incorporated
 */
```

Après les commentaires de documentation, déclarez la classe. Les commentaires d'implémentation doivent suivre directement la déclaration.

REMARQUE	N'incluez pas les commentaires non directement relatifs à la classe en cours de lecture. Par exemple, n'incluez pas de commentaires décrivant le package correspondant.

Servez-vous de blocs de commentaires, de commentaires sur une seule ligne et de commentaires de fin de ligne dans le corps de votre classe pour commenter votre code ActionScript. Pour plus d'informations sur l'utilisation des commentaires dans les fichiers de classe, consultez la section « Insertion de commentaires dans les classes », à la page 730.

Conventions de programmation ActionScript

L'un des aspects les plus importants de la programmation est la cohérence, que ce soit en matière d'appellation des variables (consultez la section « Conventions d'appellation », à la page 717), de mise en forme du code (consultez la section « Mise en forme de la syntaxe ActionScript », à la page 750) ou de normes de programmation et de placement du code ActionScript 2.0 présentés dans cette section. Le débogage du code et sa maintenance sont considérablement simplifiés lorsque le code est organisé conformément aux normes en vigueur.

Pour plus d'informations sur les conventions de programmation, consultez les sections suivantes :

Stockage du code ActionScript en un seul emplacement

Dans la mesure du possible, placez toujours votre code ActionScript 2.0 en un seul emplacement, par exemple dans un ou plusieurs fichiers ActionScript externes ou sur l'image 1 du scénario (placé sur le scénario, le code s'appelle un *script d'image*).

Si vous placez du code ActionScript dans un script d'image, placez-le sur la première ou la seconde image du scénario, sur un calque appelé *Actions*, correspondant au premier ou deuxième calque du scénario. Certaines personnes créent deux calques, pratique adéquate, pour qu'ActionScript sépare les fonctions. Certaines applications Flash ne placent pas toujours l'ensemble du code à un endroit unique (notamment lorsque vous utilisez des écrans ou des comportements).

En dehors de ces exceptions marginales, vous pouvez généralement centraliser l'ensemble de votre code. Le placement du code ActionScript dans un même emplacement présente les avantages suivants :

- Le code est facile à trouver dans un fichier source potentiellement complexe ;
- Le code est facile à déboguer.

Le plus difficile, lors du débogage des fichiers FLA, est de trouver le code source. Après avoir localisé le code, vous devez comprendre comment il interagit avec les autres éléments de code et avec le fichier FLA. Si vous centralisez l'ensemble du code dans une seule image, il devient beaucoup plus facile de le déboguer et de réduire les problèmes potentiels. Pour plus d'informations sur l'association de code à des objets (et la décentralisation du code), consultez la section « Association de code à des objets », à la page 732. Pour plus d'informations sur les comportements et le code décentralisé, consultez le guide *Utilisation de Flash*.

Association de code à des objets

Evitez d'associer votre ActionScript à des objets (tels que des occurrences de bouton ou de clip) dans un fichier FLA, même dans les applications simples ou les prototypes. Le fait d'associer un code à un objet signifie que vous sélectionnez un clip, un composant ou une occurrence de bouton, puis que vous ouvrez l'éditeur ActionScript (le panneau Actions ou la fenêtre de script) et ajoutez du code ActionScript avec les fonctions gestionnaires `on()` ou `onClipEvent()`.

Cette pratique est déconseillée pour les motifs suivants :

- Le code ActionScript associé aux objets est difficile à localiser et les fichiers FLA sont difficiles à modifier.
- Le code ActionScript associé aux objets est difficile à déboguer.
- Le code ActionScript écrit sur le scénario ou dans des classes est plus élégant et facile à développer.
- L'association de code ActionScript à des objets favorisent des styles de programmation médiocres.
- Le code ActionScript associé aux objets oblige les personnes en cours de formation et les lecteurs à apprendre des styles de programmation différents et de la syntaxe superflue et médiocre.
- Les utilisateurs doivent généralement réviser comment écrire les fonctions etc. sur un scénario à une date ultérieure.

Certains utilisateurs de Flash prétendent qu'il est plus simple d'apprendre ActionScript en associant le code à un objet. D'autres considèrent également qu'il est plus facile d'ajouter un code simple ou de commenter ou d'enseigner ActionScript de cette manière. Toutefois, le contraste entre les deux styles de programmation (code placé sur des objets et scripts d'image) peut dérouter les développeurs qui étudient ActionScript et devrait être évité. De plus, les utilisateurs qui apprennent comment écrire du code associé à des objets doivent rapprendre comment placer le code équivalent en tant que script d'image à une date ultérieure C'est pourquoi la cohérence présente des avantages tout au long du processus de formation, en apprenant comment écrire des scripts d'image.

L'association du code ActionScript à une bouton appelé `myBtn`apparaît comme suit. Évitez cette méthode :

```
on (release) {
  // Action.
}
```

Par contre, si vous placez le code ActionScript équivalent sur le scénario, vous obtenez ce qui suit :

```
// code correct
myBtn.onRelease = function() {
  // Action.
};
```

Pour plus d'informations sur la syntaxe d'ActionScript, consultez la section « Mise en forme de la syntaxe ActionScript », à la page 750.

REMARQUE

L'utilisation de comportements et d'écrans impliquant parfois l'association du code à des objets, les consignes diffèrent si vous utilisez ces fonctionnalités Pour plus d'informations, consultez le guide *Utilisation de Flash*.

Gestion des domaines

Le domaine correspond à la zone où la variable est connue et peut être utilisée dans un fichier SWF, comme sur un scénario, pouvant s'appliquer à l'échelle de l'application, ou localement au sein d'une fonction. Il existe généralement plusieurs moyens de référencer le domaine pendant l'écriture du code. Un bon usage du domaine signifie que votre code ActionScript est portable et réutilisable, sans risque de séparer vos applications au fur et à mesure du développement de nouveaux modules.

Il est important de comprendre la différence entre les deux domaines, global et racine. Le domaine racine est unique pour chaque fichier SWF chargé. Le domaine global s'applique à l'ensemble des scénarios et des domaines au sein des fichiers SWF. Appliquez un adressage relatif plutôt que des références aux scénarios racine, car le code devient alors recyclable et portable. Pour plus d'informations sur la gestion de domaine dans vos applications, consultez les sections suivantes :

« Variables et domaine », à la page 62

« Domaine et ciblage », à la page 91

« Distinction entre classes et domaine », à la page 261

Utilisation restreinte des cibles absolues (_root)

Vous pouvez cibler des occurrences de plusieurs façons pour éviter d'utiliser `_root` ; ces méthodes sont présentées plus bas dans cette section. Evitez d'utiliser `_root` dans du code ActionScript 2.0, car cela risque d'interférer avec le fonctionnement des fichiers SWF chargés par d'autres fichiers du même type. L'identificateur `_root` cible le fichier SWF de base en cours de chargement et non le fichier SWF utilisant un adressage relatif à la place de `_root`. Ce problème limite la portabilité du code dans les fichiers SWF qui sont chargés dans un autre fichier, et, plus particulièrement, dans des composants et des clips. Vous pouvez résoudre certains problèmes avec `_lockroot`, mais utilisez uniquement `_lockroot` lorsque c'est nécessaire (comme lorsque vous chargez un fichier SWF sans avoir accès au fichier FLA). Pour plus d'informations sur l'utilisation de `_lockroot`, consultez la section « Utilisation de _lockroot », à la page 735.

Employez les mots-clés `this`, `this._parent` ou `_parent` de préférence à `_root`, selon l'emplacement du code ActionScript 2.0. L'exemple suivant illustre l'adressage relatif :

```
myClip.onRelease = function() {
  trace(this._parent.myButton._x);
};
```

Vous devez limiter le domaine des variables, sauf pour les variables qui servent de paramètres de fonction et pour les variables locales. Limitez les domaines des variables à leur chemin actuel dans la mesure du possible, en employant un adressage relatif, tel que la propriété `this`. Pour plus d'informations sur l'utilisation de la propriété `this`, consultez la section propriété this dans le *Guide de référence du langage ActionScript 2.0*.

Utilisation de _lockroot

L'emploi de `_lockroot` pour cibler un contenu permet de résoudre les problèmes de domaine parfois liés à l'utilisation inappropriée de `_root`. Bien que cela résolve de nombreux problèmes dans les applications, utilisez `_lockroot` comme méthode de contournement pour les problèmes générés par `_root`. En cas de problème lors du chargement de contenu dans un fichier SWF ou une occurrence de composant, tentez d'appliquer `_lockroot` à un clip qui charge le contenu. Par exemple, si le clip appelé `myClip` charge du contenu et cesse de fonctionner à la fin du chargement, tentez d'utiliser le code suivant placé parfois sur un scénario :

```
this._lockroot = true;
```

Utilisation du mot-clé this

Dans la mesure du possible, utilisez le mot-clé `this` en tant que préfixe au lieu d'omettre le mot-clé, même si votre code peut s'en passer. Utilisez le mot-clé `this` pour déterminer si une méthode ou une propriété appartient à une classe spécifique. Par exemple, pour une fonction sur le scénario principal, écrivez du code ActionScript 2.0 au format suivant :

```
circleClip.onPress = function() {
  this.startDrag();
};
circleClip.onRelease = function() {
  this.stopDrag();
};
```

Pour une classe, écrivez du code au format suivant :

```
class User {
  private var username:String;
  private var password:String;
  function User(username:String, password:String) {
    this.username = username;
    this.password = password;
  }
  public function get username():String {
    return this.username;
  }
  public function set username(username:String):Void {
    this.username = username;
  }
}
```

Si vous ajoutez systématiquement le mot-clé `this` dans ces situations, le code ActionScript 2.0 devient beaucoup plus facile à lire et comprendre.

Présentation des domaines dans les classes

Lorsque vous portez du code dans les classes ActionScript 2.0, il peut être nécessaire de changer le mode d'utilisation du mot clé `this`. Par exemple, si une méthode de classe utilise une fonction de rappel (telle que la méthode `onLoad` de la classe LoadVars), il sera difficile de savoir si le mot-clé `this` fait référence à la classe ou à l'objet LoadVars. Dans ce cas, il peut être nécessaire de créer un pointeur vers la classe actuelle, comme dans l'exemple suivant :

```
class Product {
  private var m_products_xml:XML;
  // Constructeur
  // targetXml_string contient le chemin vers un fichier XML
  function Product(targetXmlStr:String) {
    /* Crée une référence locale à la classe actuelle.
       Même si vous êtes placé au niveau du gestionnaire d'événement onLoad
  du code XML,
       vous pouvez référencer la classe actuelle et pas seulement le paquet
  XML. */
    var thisObj:Product = this;
    // Crée une variable locale, qui permet de charger le fichier XML.
    var prodXml:XML = new XML();
    prodXml.ignoreWhite = true;
    prodXml.onLoad = function(success:Boolean) {
      if (success) {
        /* Lorsque le code XML se charge et est analysé correctement,
           définir la variable m_products_xml de la classe sur le document
           XML analysé et appeler la fonction init. */
        thisObj.m_products_xml = this;
        thisObj.init();
      } else {
        /* Une erreur s'est produite pendant le chargement du fichier XML.
  */
        trace("error loading XML");
      }
    };
    // Amorce du chargement du document XML
    prodXml.load(targetXmlStr);
  }
  public function init():Void {
    // Affichage du paquet XML
    trace(this.m_products_xml);
  }
}
```

Dans la mesure où vous tentez de référencer la variable du membre privé au sein d'un gestionnaire `onLoad`, le mot-clé `this` fait en réalité référence à l'occurrence `prodXml` et non à la classe `Product`, à laquelle on s'attendrait. Par conséquent, vous devez créer un pointeur vers le fichier de classe local de façon à pouvoir référencer directement la classe à partir du gestionnaire `onLoad`.

Pour plus d'informations sur les classes, consultez le « Distinction entre classes et domaine », à la page 261. Pour plus d'informations sur le domaine, consultez la section « Gestion des domaines », à la page 733.

Structure d'un fichier de classe

Vous créez des classes dans des fichiers ActionScript 2.0 distincts et qui sont importés dans un fichier SWF au moment de sa compilation.

Vous créez des classes dans des fichiers ActionScript 2.0 distincts, importés dans un fichier SWF lors de la compilation de l'application. Pour créer un fichier de classe, vous écrivez du code avec une certaine méthodologie et un certain ordre. Cette méthodologie est présentée dans les sections suivantes :

Les conventions suivantes de structure d'un fichier de classe décrivent l'organisation possible des parties d'une classe en vue de l'amélioration de l'efficacité et de la lisibilité du code.

Pour structurer un fichier de classe, utilisez les éléments suivants :

1. Ajoutez des commentaires de documentation qui incluent une description générale du code, en plus des informations sur l'auteur et la version.
2. Ajoutez vos instructions d'importation (le cas échéant).
3. Ecrivez une déclaration de classe ou d'interface comme suit :

   ```
   ClasseUtilisateur{...}
   ```
4. Incluez tout commentaire d'implémentation de classe ou d'interface nécessaire.

   Dans ces commentaires, ajoutez des informations pertinentes pour l'ensemble de la classe ou de l'interface.
5. Ajoutez toutes vos variables statiques.

   Ecrivez les variables de classe publiques en premier, suivies des variables de classe privées.
6. Ajoutez des variables d'occurrence.

   Ecrivez les variables de membre publiques en premier, suivies des variables de membre privées.
7. Ajoutez l'instruction constructeur, par exemple celle de l'exemple suivant :

   ```
   public function UserClass(username:String, password:String) {...}
   ```
8. Ecrivez vos méthodes.

   Regroupez les méthodes selon leur fonctionnalité et non pas leur accessibilité ou domaine. En organisant les méthodes de cette manière, vous améliorez la lisibilité et la clarté de votre code.
9. Ecrivez les méthodes de lecture/définition dans le fichier de classe.

Directives sur la création d'une classe

Lorsque vous créez un fichier de classe, respectez les consignes suivantes :

- Ne placez pas plusieurs déclarations sur une seule ligne.

  Par exemple, formatez les accolades comme dans l'exemple suivant :

  ```
  var prodSkuNum:Number;    // numéro de stock du produit
                            // (numéro d'identification)
  var prodQuantityNum:Number; // Quantité du produit
  ```

  Cet exemple illustre comment placer ces déclarations. Placez ces déclarations au début d'un bloc de code.

- Initialisez les variables locales quand elles ont été déclarées.

  Les propriétés d'une classe doivent être initialisées dans la déclaration si l'initialiseur est une constante chargée à la compilation.

- Déclarez les variables avant de les utiliser.

  Ceci comprend les boucles.

- Evitez d'utiliser des déclarations locales qui masquent des déclarations de niveau supérieur.

  Par exemple, ne déclarez une variable qu'une seule fois, comme dans l'exemple ci-dessous :

  ```
  var counterNum:Number = 0;
  function myMethod() {
    for (var counterNum:Number = 0; counterNum<=4; counterNum++) {
      // instructions ;
    }
  }
  ```

  Cet exemple de code déclare la même variable au sein d'un bloc interne, ce qui doit être évité.

- N'assignez pas beaucoup de variables à une seule valeur dans une instruction.

  Respectez cette convention, sans quoi votre code sera difficile à lire, comme le montre le code ActionScript suivant :

  ```
  playBtn.onRelease = playBtn.onRollOut = playsound;
  ```

  ou

  ```
  class User {
    private var m_username:String, m_password:String;
  }
  ```

- Ne rendez publique une méthode ou une propriété que s'il y a une raison pour ce faire. Autrement, rendez privées vos méthodes et propriétés.

- N'*abusez* pas des fonctions de lecture/définition dans le fichier de classe.

  Les fonctions lecture/définition sont excellentes pour toute une variété d'utilisations (consultez la section « Présentation des méthodes de lecture et définition », à la page 229). Toutefois, une utilisation abusive peut indiquer que vous pourriez améliorer l'architecture ou l'organisation de votre application.
- Définissez la plupart des variables membres en mode privé à moins que vous ayiez une bonne raison de les rendre publiques.

  D'un point de vue de conception, il est préférable de rendre privées les variables membres et d'autoriser l'accès à ces variables uniquement à travers un groupe de fonctions lecture/définition.

Utilisation du préfixe this dans les fichiers de classe

Utilisez le mot-clé `this` comme un préfixe au sein des classes pour les méthodes et les variables membres Bien que ce ne soit pas nécessaire, il est plus facile de déterminer si une propriété ou une méthode appartient à une classe lorsqu'elle possède un préfixe ; sans quoi, il n'est pas possible de déterminer si la propriété ou la méthode appartient à la superclasse.

Vous pouvez également utiliser un préfixe de nom de classe pour les variables statiques et les méthodes, y compris dans une classe. Vos références sont ainsi plus claires. La qualification des références rend le code plus lisible. Selon votre environnement de programmation, l'utilisation de préfixes permet également de bénéficier de la saisie automatique et des conseils de code. Le code suivant illustre l'attribution d'un préfixe à une propriété statique avec un nom de classe :

```
class Widget {
  public static var widgetCount:Number = 0;
  public function Widget() {
    Widget.widgetCount++;
  }
}
```

REMARQUE

Ces préfixes sont facultatifs et certains développeurs les considèrent superflus. Adobe vous recommande d'ajouter le mot-clé `this` en tant que préfixe, parce qu'il peut améliorer la lisibilité et vous aider à écrire un code propre en fournissant le contexte.

Présentation de l'initialisation

Pour les valeurs initiales des variables, assignez une valeur par défaut ou autorisez la valeur de `indéfinie`, comme illustré dans l'exemple de classe suivant Lorsque vous initialisez les propriétés incorporées, l'expression à droite d'une commande doit être une constante chargée à la compilation. C'est-à-dire que l'expression ne peut référer à quoi que ce soit de défini en cours d'exécution. Les constantes chargées à la compilation comprennent des littérals chaînes, nombres, valeurs booléennes, zéro et indéfinies, ainsi que des fonctions de contructeur pour les classes de haut niveau suivantes : Array, Boolean, Number, Object et String. Cette classe définit les valeurs initiales de `m_username` et `m_password` aux chaînes vides :

```
class User {
  private var m_username:String = "";
  private var m_password:String = "";
  function User(username:String, password:String) {
    this.m_username = username;
    this.m_password = password;
  }
}
```

Supprimez des variables ou rendez des variables `nulles` quand vous n'en avez plus besoin. Le réglage des variables sur `nulles` peut améliorer la performance. Ce processus est communément appelé *récupérage de place*. La suppression des variables permet d'optimiser la mémoire pendant l'exécution, dans la mesure où les ressources inutiles sont supprimées du fichier SWF. Il vaut mieux supprimer les variables que de les définir sur `nul`. Pour plus d'informations sur les performances, consultez la section « Optimisation du code », à la page 749.

Pour plus d'informations sur l'appellation des variables, consultez la section « Appellation des variables », à la page 720. Pour plus d'informations sur la suppression d'objets, consultez la section relative à l'instruction delete dans le *Guide de référence du langage ActionScript 2.0*

L'utilisation des classes est l'un des moyens les plus simples d'initialiser le code à l'aide d'ActionScript 2.0. Vous pouvez capsuler toute votre initialisation pour une occurrence au sein de la fonction de constructeur de la classe, ou l'astraire dans une méthode séparée, que vous appelleriez explicitement après création de la variable, comme l'illustre le code suivant :

```
class Product {
  function Product() {
    var prodXml:XML = new XML();
    prodXml.ignoreWhite = true;
    prodXml.onLoad = function(success:Boolean) {
      if (success) {
        trace("loaded");
      } else {
        trace("error loading XML");
      }
    };
    prodXml.load("products.xml");
  }
}
```

Le code suivant pourrait être le premier appel de fonction dans l'application, et le seul que vous faites pour l'initialisation. La première image d'un fichier FLA qui est en train de charger du XML peut utiliser un code similaire à l'ActionScript suivant :

```
if (init == undefined) {
  var prodXml:XML = new XML();
  prodXml.ignoreWhite = true;
  prodXml.onLoad = function(success:Boolean) {
    if (success) {
      trace("loaded");
    } else {
      trace("error loading XML");
    }
  };
  prodXml.load("products.xml");
  init = true;
}
```

Utilisation des instructions de traçage

Utilisez les instructions de `traçage` dans vos documents pour vous aider à déboguer votre code tout en autorisant le fichier FLA Par exemple, en utilisant une instruction de `traçage` et la boucle `for`, vous pouvez voir les valeurs des variables dans le panneau de sortie, comme les chaînes, les tableaux et les objets, comme le montre l'exemple suivant :

```
var dayArr:Array = ["sun", "mon", "tue", "wed", "thu", "fri", "sat"];
var numOfDays:Number = dayArr.length;
for (var i = 0; i<numOfDays; i++) {
  trace(i+": "+dayArr[i]);
}
```

Ceci affiche les informations suivantes dans le panneau de sortie :

```
0: sun
1: mon
2: tue
3: wed
4: thu
5: fri
6: sat
```

L'utilisation d'une instruction de `traçage` est un moyen efficace de déboguer votre ActionScript 2.0.

Vous pouvez retirer vos instructions de `traçage`en publiant un fichier SWF, ce qui fait de légères améliorations à la lecture Avant de publier un fichier SWF, ouvrez Réglages de Publication et sélectionnez Omit Trace Actions sur le Flash tab. Pour plus d'informations sur l'utilisation d'un traçage, consultez la section relative à la fonction de traçage dans le *Guide de référence du langage ActionScript 2.0*

A propos du préfixe super

Si vous vous référez à une méthode dans la classe parent, attribuez un préfixe à la méthode avec `super` afin que les autres développeurs sachent d'où la méthode provient L'extrait d'ActionScript 2.0 suivant illustre l'utilisation d'un domaine approprié en employant le préfixe `super` :

Dans l'exemple qui suit, vous créez deux classes L'utilisation du mot-clé `super` dans la classe Chaussettes permet d'appeler des fonctions dans la classe parent (Habits) Bien que les classes des Socks et Clothes possèdent une méthode appelée `getColor()`, en employant `super` vous permet de vous référer en particulier aux méthodes et propriétés de la classe de base Créez un nouveau fichier AS nommé Clothes.as, et entrez le code suivant :

```
class Clothes {
   private var color:String;
   function Clothes(paramColor) {
      this.color = paramColor;
      trace("[Clothes] I am the constructor");
   }
   function getColor():String {
      trace("[Clothes] I am getColor");
      return this.color;
   }
   function setColor(paramColor:String):Void {
      this.color = paramColor;
      trace("[Clothes] I am setColor");
   }
}
```

Créez une nouvelle classe appelée Socks qui provient de la classe Clothes, comme l'illustre l'exemple suivant :

```
class Socks extends Clothes {
   private var color:String;
   function Socks(paramColor:String) {
      this.color = paramColor;
      trace("[Socks] I am the constructor");
   }
   function getColor():String {
      trace("[Socks] I am getColor");
      return super.getColor();
   }
   function setColor(paramColor:String):Void {
      this.color = paramColor;
      trace("[Socks] I am setColor");
   }
}
```

Créez ensuite un nouveau fichier AS ou FLA et entrez l'ActionScript suivant dans le document :

```
import Socks;
var mySock:Socks = new Socks("maroon");
trace(" -> "+mySock.getColor());
mySock.setColor("Orange");
trace(" -> "+mySock.getColor());
```

Ce qui suit s'affiche dans le panneau de sortie :

```
[Clothes] I am the constructor
[Socks] I am the constructor
[Socks] I am getColor
[Clothes] I am getColor
-> maroon
[Socks] I am setColor
[Socks] I am getColor
[Clothes] I am getColor
-> Orange
```

Si vous avez oublié de mettre le mot-clé `super` dans la méthode `getColor()` de la classe Socks, la méthode `getColor()` pourrait s'appeler de manière répétitive, ce qui entraînerait des erreurs dans le script en raison de problèmes récurrents à l'infini. Si vous n'utilisiez pas le mot-clé super, le panneau de sortie afficherait l'erreur suivante :

```
[Socks] I am getColor
[Socks] I am getColor
...
[Socks] I am getColor
256 levels of recursion were exceeded in one action list.
This is probably an infinite loop.
Further execution of actions has been disabled in this SWF file.
```

Utilisation restreinte de l'instruction with

L'utilisation de l'instruction `with` est l'un des concepts qui induisent le plus en erreur les personnes qui apprennent ActionScript 2.0 Tenez compte du code suivant qui utilise l'instruction `with` :

```
this.attachMovie("circleClip", "circle1Clip", 1);
with (circle1Clip) {
  _x = 20;
  _y = Math.round(Math.random()*20);
  _alpha = 15;
  createTextField("labelTxt", 100, 0, 20, 100, 22);
  labelTxt.text = "Circle 1";
  someVariable = true;
}
```

Dans ce code, vous allez associer une occurrence de clip depuis la bibliothèque et utiliser l'instruction `with` pour modifier ses propriétés. Vous devez spécifier le domaine d'une variable pour savoir où définir les propriétés, faute de quoi le code risque de paraître confus. Dans le code précédent, on pourrait s'attendre à ce que `une Variable` soit définie au sein du clip `circle1Clip`, mais en réalité elle est définie dans un scénario du fichier SWF.

Il est plus facile de suivre ce qui se passe dans le code si vous précisez explicitement le domaine des variables, au lieu de compter sur l'instruction `with`. L'exemple suivant présente un exemple de code ActionScript spécifiant le domaine des variables qui est légèrement plus long, mais préférable.

```
this.attachMovie("circleClip", "circle1Clip", 1);
circle1Clip._x = 20;
circle1Clip._y = Math.round(Math.random()*20);
circle1Clip._alpha = 15;
circle1Clip.createTextField("labelTxt", 100, 0, 20, 100, 22);
circle1Clip.labelTxt.text = "Circle 1";
circle1Clip.someVariable = true;
```

Une seule exception à cette règle est quand vous travaillez avec le dessin API pour dessiner des formes, il se peut que vous ayiez des appels similaires aux mêmes méthodes (comme `lineTo` ou `curveTo`) en raison de la fonctionnalité de l'outil dessin API. Par exemple, quand vous dessinez un simple rectangle, il faut quatre appels séparés à la méthode `lineTo`, comme l'illustre le code suivant :

```
this.createEmptyMovieClip("rectangleClip", 1);
with (rectangleClip) {
  lineStyle(2, 0x000000, 100);
  beginFill(0xFF0000, 100);
  moveTo(0, 0);
  lineTo(300, 0);
  lineTo(300, 200);
  lineTo(0, 200);
  lineTo(0, 0);
  endFill();
}
```

Si vous écriviez chaque méthode `lineTo` ou `curveTo` avec un nom d'occurrence qualifié, le code deviendrait rapidement surchargé et difficile à lire et déboguer.

Utilisation des fonctions

Recyclez des blocs de code dans la mesure du possible. Pour ce faire, vous pouvez appeler une fonction plusieurs fois, au lieu de créer du code de toutes pièces à chaque fois. Les fonctions pouvant comporter du code générique, il est donc possible d'utiliser les mêmes blocs à des fins légèrement différentes dans un fichier SWF. Le recyclage du code permet de créer des applications efficaces et de réduire la quantité de code ActionScript 2.0 nécessaire, et donc la durée du développement. Vous pouvez créer des fonctions sur un scénario, dans un fichier de classe ou écrire du code ActionScript qui réside dans un composant de type code et les réutiliser de diverses manières.

Si vous utilisez ActionScript 2.0, ne placez pas de fonctions sur le scénario. Lorsque vous utilisez ActionScript 2.0, placez des fonctions dans des fichiers de classe chaque fois que possible, comme dans l'exemple suivant :

```
class Circle {
public function area(radius:Number):Number {
  return (Math.PI*Math.pow(radius, 2));
}
public function perimeter(radius:Number):Number {
  return (2 * Math.PI * radius);
}
public function diameter(radius:Number):Number {
  return (radius * 2);
}
}
```

Appliquez la syntaxe suivante lorsque vous créez des fonctions :

```
function myCircle(radius:Number):Number {
  //...
}
```

Evitez d'utiliser la syntaxe suivante, qui est difficile à lire :

```
myCircle = function(radius:Number):Number {
  //...
}
```

L'exemple suivant place des fonctions dans un fichier de classe. Il s'agit de la meilleure pratique lorsque vous choisissez d'utiliser le code ActionScript 2.0, dans la mesure où elle optimise le recyclage du code. Lorsque vous devez utiliser de nouveau les fonctions dans d'autres applications, vous pouvez importer la classe existante sans avoir à ré-écrire le code de toutes pièces ou dupliquer les fonctions dans la nouvelle application.

```
class mx.site.Utils {
  static function randomRange(min:Number, max:Number):Number {
    if (min>max) {
      var temp:Number = min;
      min = max;
      max = temp;
    }
    return (Math.floor(Math.random()*(max-min+1))+min);
  }
  static function arrayMin(numArr:Array):Number {
    if (numArr.length == 0) {
      return Number.NaN;
    }
    numArr.sort(Array.NUMERIC | Array.DESCENDING);
    var min:Number = Number(numArr.pop());
    return min;
  }
  static function arrayMax(numArr:Array):Number {
    if (numArr.length == 0) {
      return undefined;
    }
    numArr.sort(Array.NUMERIC);
    var max:Number = Number(numArr.pop());
    return max;
  }
}
```

Vous pouvez utiliser ces fonctions en ajoutant le code ActionScript suivant à votre fichier FLA :

```
import mx.site.Utils;
var randomMonth:Number = Utils.randomRange(0, 11);
var min:Number = Utils.arrayMin([3, 3, 5, 34, 2, 1, 1, -3]);
var max:Number = Utils.arrayMax([3, 3, 5, 34, 2, 1, 1, -3]);
trace("month: "+randomMonth);
trace("min: "+min);
trace("max: "+max);
```

À propos de l'arrêt de la répétition de code

Le gestionnaire d'événement `onEnterFrame` est utile car il permet à Flash de répéter du code à la cadence d'images d'un fichier SWF. Cependant, limitez la quantité de répétition que vous utilisez dans un fichier Flash autant que possible de façon à préserver les performances. Par exemple, si un élément de code se répète lorsque la tête de lecture passe sur une image, le processeur est fortement sollicité. Ceci risque d'entraîner des problèmes de performances sur les ordinateurs qui lisent le fichier SWF. Si vous utilisez le gestionnaire d'événement `onEnterFrame` pour tout type d'animation ou de répétition dans vos fichiers SWF, supprimez le gestionnaire `onEnterFrame` après l'avoir utilisé. Dans le code ActionScript 2.0 suivant, vous arrêtez la répétition en supprimant le gestionnaire d'événements `onEnterFrame` :

```
circleClip.onEnterFrame = function() {
  circleClip._alpha -= 5;
  if (circleClip._alpha<=0) {
    circleClip.unloadMovie();
    delete this.onEnterFrame;
    trace("deleted onEnterFrame");
  }
};
```

De même, limitez l'utilisation de `setInterval` et pensez à supprimer l'intervalle lorsque vous avez terminé de l'utiliser pour réduire la sollicitation du processeur par le fichier SWF.

Optimisation d'ActionScript et de Flash Player

Si vous compilez un fichier SWF contenant du code ActionScript 2.0 dont l'option Paramètres de publication est définie sur Flash Player 6 et ActionScript 1.0, le code fonctionne tant qu'il n'utilise pas les classes ActionScript 2.0. Le code ne respecte pas la casse, seul Flash Player la respecte. Cependant, si vous compilez votre fichier SWF avec l'option Paramètres de publication définie sur Flash Player 7 ou 8 et ActionScript 1.0, Flash impose le respect de la casse.

Annotations de type de données (types de données strictes) sont imposées sur un au moment de la compilation pour Flash Player 7 et versions ultérieures quand les paramètres de publication sont définis sur ActionScript 2.0.

ActionScript 2.0 compile vers le pseudo-code ActionScript 1.0 lorsque vous publiez vos applications, et vous pouvez ainsi cibler Flash Player 6 et versions ultérieures lorsque vous travaillez avec ActionScript 2.0.

Pour plus d'informations sur l'optimisation de vos applications, consultez « Optimisation du code ».

Optimisation du code

Lorsque vous optimisez votre code, respectez les consignes suivantes :

- Evitez d'appeler la même fonction plusieurs fois à partir d'une boucle.

  Il est préférable d'inclure le contenu d'une petite fonction dans la boucle.
- Utilisez des fonctions exécutables quand c'est possible.

  Les fonctions exécutables sont plus rapides que les fonctions définies par les utilisateurs.
- N'abusez pas du type objet

  Les annotations de type de données doivent être précises pour améliorer les performances. N'utilisez un type d'objet que lorsqu'il n'y a pas d'autre possibilité raisonnable.
- Evitez la fonction `eval()` ou l'opérateur d'accès au tableau.

  Souvent, régler une fois la référence locale est préférable et plus efficace.
- Affectez `Array.length` à une variable avant la boucle.

  Affectez `Array.length` à une variable avant la boucle pour l'utiliser tel quel, plutôt que d'utiliser `myArr.length` lui-même Par exemple,

  ```
  var fontArr:Array = TextField.getFontList();
  var arrayLen:Number = fontArr.length;
  for (var i:Number = 0; i < arrayLen; i++) {
    trace(fontArr[i]);
  }
  ```

  Au lieu de :

  ```
  var fontArr:Array = TextField.getFontList();
  for (var i:Number = 0; i < fontArr.length; i++) {
    trace(fontArr[i]);
  }
  ```
- Concentrez-vous sur l'optimisation des boucles et sur toutes actions répétées.

  Flash Player passe beaucoup de temps à traiter des boucles (comme celles qui utilisent la fonction `setInterval()`).
- Ajoutez le mot-clé `var` lorsque vous déclarez une variable
- N'utilisez pas de variables de classe ou globales lorsque des variables locales suffisent

Mise en forme de la syntaxe ActionScript

La mise en forme standard du code ActionScript 2.0 est essentielle lorsque l'on souhaite écrire un code facile à maintenir et que les autres développeurs peuvent aisément comprendre et modifier. Par exemple, il est extrêmement difficile de suivre la logique d'un fichier FLA qui ne comporte aucun retrait ou commentaire, ou dont le formatage et les conventions d'appellation sont incohérents. Lorsque certains blocs de code (tels que les boucles et les instructions `if`) sont en retrait, le code est plus facile à lire et à déboguer.

Pour plus d'informations sur le formatage du code, consultez les rubriques suivantes :

- « Recommandations générales de formatage », à la page 750
- « Ecriture d'instructions conditionnelles », à la page 753
- « Ecriture d'instructions composées », à la page 755
- « Rédaction de l'instruction for », à la page 756
- « Rédaction des instructions while et do..while », à la page 756
- « Rédaction d'instructions return », à la page 756
- « Ecriture d'instructions switch », à la page 757
- « Écriture des instructions try..catch et try..catch..finally », à la page 757
- « Syntaxe des écouteurs », à la page 758

Recommandations générales de formatage

Utilisez des espaces, des sauts de lignes et des tabulations pour rendre votre code plus facile à lire. L'ajout d'espaces améliore la clarté car elle met en évidence et renforce la hiérarchisation du code. L'espacement et la lisibilité sont des facteurs importants car ils rendent le code ActionScript2.0 plus facile à comprendre, point capital pour les stagiaires, mais également pour les utilisateurs expérimentés qui travaillent sur des projets complexes. La lisibilité est importante lorsque vous déboguez du code ActionScript, car il est beaucoup plus facile de détecter des erreurs lorsque le code est formaté et espacé correctement.

Le code ActionScript 2.0 peut être écrit et mis en forme de plusieurs façons. La syntaxe peut présenter des différences, telles que la présentation sur plusieurs lignes dans l'éditeur ActionScript (panneau Action ou fenêtre de script), l'emplacement des accolades (`{}`) ou des parenthèses [`()`]).

Les conseils de formatage suivants permettent d'accroître la lisibilité du code ActionScript.

- Laissez une ligne vide entre les paragraphes (modules) d'ActionScript.

  Les paragraphes de codes d'ActionScript sont des groupes de codes liés de manière logique Leur espacement par une ligne vide facilite leur lecture par les utilisateurs, qui comprennent mieux la logique mise en oeuvre.

- Ménagez des indentations de dimensions identiques dans votre code. Les indentations mettent en évidence la hiérarchie de la structure du code.

  Appliquez le même style d'indentation à l'ensemble du code ActionScript et assurez-vous que les accolades ({}) sont alignées correctement. Ceci permet d'améliorer la lisibilité du code. Lorsque la syntaxe ActionScript est correcte, Flash insère automatiquement des indentations lorsque vous appuyez sur la touche Entrée (Windows) ou Retour (Macintosh). Vous pouvez également cliquer sur le bouton Format automatique de l'éditeur ActionScript (panneau Actions ou fenêtre de script) pour insérer des indentations lorsque la syntaxe ActionScript est correcte.

- Utilisez des sauts de ligne pour rédiger des instructions complexes plus faciles à lire.

  Vous pouvez formater certaines instructions, comme des instructions conditionnelles. Vous disposez pour cela de plusieurs méthodes : Parfois, le simple fait de répartir les instructions sur plusieurs lignes suffit pour faciliter leur lecture.

- Insérez un espace après un mot clé qui est suivi par des parenthèses [()].

  Le code ActionScript suivant en est un exemple :

```
do {
  //quelque chose
} while (condition);
```

- Ne laissez pas d'espace entre un nom de méthode et des parenthèses.

  Le code ActionScript suivant en est un exemple :

```
function checkLogin():Boolean {
  // instructions ;
}
checkLogin();
```

  ou

```
printSize("size is " + foo + "\n");
```

- Insérez un espace après les virgules dans une liste d'instructions.

  L'utilisation des espaces après les virgules simplifie la distinction entre les appels de méthode et les mot clés, comme illustré dans l'exemple suivant :

```
function addItems(item1:Number, item2:Number):Number {
  return (item1 + item2);
}
var sum:Number = addItems(1, 3);
```

- Utilisez les espaces pour séparer tous les opérateurs et leurs opérandes.

  L'utilisation des espaces simplifie la distinction entre les appels de méthode et les mot clés, comme illustré dans l'exemple suivant :

  ```
  // Correct
  var sum:Number = 7 + 3;
  // Incorrect
  var sum:Number=7+3;
  ```

  L'une des exceptions à cette directive est l'opérateur point (.).

- Ne laissez pas d'espace entre les opérateurs et leur associativité.

  Par exemple, incrémentez (++) et décrémentez (--) comme dans l'exemple suivant :

  ```
  while (d++ = s++)
  -2, -1, 0
  ```

- Ne laissez pas d'espace après ouverture d'une parenthèse et avant fermeture une parenthèse.

  Le code ActionScript suivant en est un exemple :

  ```
  // Incorrect
  ( "size is " + foo + "\n" );
  // Correct
  ("size is " + foo + "\n");
  ```

- Ne placez qu'une seule instruction par ligne pour améliorer la lisibilité de votre code ActionScript.

  Le code ActionScript suivant en est un exemple :

  ```
  theNum++;       // Correct.
  theOtherNum++;  // Correct.
  aNum++; anOtherNum++;  // Incorrect
  ```

- N'incorporez par les instructions.

  Les instructions imbriquées sont parfois employées pour améliorer les performances d'un fichier SWF lors de l'exécution, mais le code s'en trouve bien plus difficile à lire et à déboguer. Le code ActionScript suivant illustre cette mise en forme (mais n'oubliez pas d'éviter les noms à un seul caractère dans le code réel) :

  ```
  var myNum:Number = (a = b + c) + d;
  ```

- Affectez les variables en tant qu'instructions distinctes.

  Le code ActionScript suivant illustre cette mise en forme (mais n'oubliez pas d'éviter les noms à un seul caractère dans le code réel) :

  ```
  var a:Number = b + c;
  var myNum:Number = a + d;
  ```

- Revenez à la ligne avant un opérateur.
- Revenez à la ligne après une virgule.

- Alignez la deuxième ligne avec le début de l'expression de la ligne de code précédente.

REMARQUE	Vous pouvez contrôler les paramètres d'indentation automatiques et manuels en sélectionnant Modifier › Préférences (Windows) ou Flash › Préférences (Macintosh), puis en sélectionnant l'onglet ActionScript.

Ecriture d'instructions conditionnelles

Respectez les consignes suivantes lorsque vous rédigez des instructions conditionnelles :

- Placez les conditions sur des lignes distinctes dans des instructions `if`, `else..if` et `if.else`.
- Utilisez les accolades (`{}`) pour les instructions `if`.
- Formatez les accolades comme dans les exemples suivants :

```
// instruction if
if (condition) {
  // instructions
}

// Instruction if..else
if (condition) {
  // instructions
} else {
  // instructions
}

// Instruction else..if
if (condition) {
  // instructions
} else if (condition) {
  // instructions
} else {
  // instructions
}
```

Lorsque vous écrivez des conditions complexes, il est préférable de les regrouper entre parenthèses [()]. Si vous omettez les parenthèses, vous (ou toute autre personne travaillant sur votre code ActionScript 2.0) risquez de rencontrer des erreurs au niveau de la priorité des opérateurs.

Par exemple, le code suivant ne met pas la condition entre parenthèses :

```
if (fruit == apple && veggie == leek) {}
```

Le code suivant met les conditions entre parenthèses de manière appropriée :

```
if ((fruit == apple) && (veggie == leek)) {}
```

Vous pouvez écrire une instruction conditionnelle renvoyant une valeur booléenne de deux manières. Le second exemple est préférable :

```
if (cartArr.length>0) {
  return true;
} else {
  return false;
}
```

Comparez cet exemple avec le précédent :

```
// Préférable
return (cartArr.length > 0);
```

Le deuxième fragment de code est plus concis et comporte moins d'expressions à évaluer. Il est plus facile à lire et à comprendre.

L'exemple suivant permet de vérifier si la variable *y* est supérieure à zéro (0), ce qui renvoie le résultat `x/y` ou la valeur zéro (0).

```
return ((y > 0) ? x/y : 0);
```

L'exemple suivant propose une autre possibilité de rédaction de ce code : Le second exemple est préférable :

```
if (y>0) {
  return x/y;
} else {
  return 0;
}
```

La syntaxe abrégée de l'instruction `if` du premier exemple est appelée opérateur conditionnel (`?:`). Il permet de convertir des instructions `if-else` simples en une ligne de code. Dans ce cas, la syntaxe abrégée nuit à la lisibilité.

Si les opérateurs conditionnels sont vraiment incontournables, placez la première condition (avant le point d'interrogation [`?`]) entre parenthèses pour améliorer la lisibilité du code. Le code précédent était un exemple de cette mise en forme.

Ecriture d'instructions composées

Les instructions composées contiennent une liste d'instructions entre accolades ({}). Ces instructions entre accolades sont en retrait par rapport à l'instruction composée, Le code ActionScript suivant en est un exemple :

```
if (a == b) {
  // Ce code est en retrait.
  trace("a == b");
}
```

Placez les instructions entre accolades lorsqu'elles appartiennent à une structure de contrôle (`if..else` ou `for`), même lorsque cette dernière ne comporte qu'une seule instruction. L'exemple suivant montre un code médiocre :

```
// Incorrect
if (numUsers == 0)
  trace("no users found.");
```

Bien que ce code puisse être validé, sa présentation est médiocre car ses instructions ne sont pas entre accolades. Dans ce cas, le fait d'ajouter une autre instruction après l'instruction `trace` permet de l'exécuter, que la variable `numUsers` soit égale à 0 ou non :

```
// Incorrect
var numUsers:Number = 5;
if (numUsers == 0)
  trace("no users found.");
  trace("I will execute");
```

L'exécution du code malgré la variable `numUsers` peut entraîner des résultats inattendus. Par conséquent, ajoutez des accolades, comme dans l'exemple suivant :

```
var numUsers:Number = 0;
if (numUsers == 0) {
  trace("no users found");
}
```

Lors de la rédaction d'une condition, n'ajoutez pas de `==true` redondant dans votre code, comme suit :

```
if (something == true) {
  // instructions
}
```

Pour une comparaison à `false`, utilisez `if (quelquechose==false)` ou `if(!quelquechose)`.

Rédaction de l'instruction for

Rédigez les instructions `for` au format suivant :

```
for (init; condition; update) {
  // instructions
}
```

La structure suivante montre la mise en forme de l'instruction `for` :

```
var i:Number;
for (var i = 0; i<4; i++) {
  myClip.duplicateMovieClip("newClip" + i + "Clip", i + 10, {_x:i*100,
  _y:0});
}
```

Vous devez inclure un espace après chaque expression d'une instruction `for`.

Rédaction des instructions while et do..while

Rédigez les instructions `while` au format suivant :

```
while (condition) {
  // instructions
}
```

Ecrivez les instructions `do-while` au format suivant :

```
do {
  // instructions
} while (condition);
```

Rédaction d'instructions return

N'utilisez pas de parenthèses [()] avec les instructions return qui comportent des valeurs. Vous ne devez utiliser de parenthèses dans les instructions `return` que lorsque cela rend la valeur plus évidente, ce qui est illustré dans la troisième ligne du code ActionScript suivant :

```
return;
return myCar.paintColor;
// les parenthèses permettent de mettre la valeur de return en évidence
return ((paintColor)? paintColor: defaultColor);
```

Ecriture d'instructions switch

- Toutes les instructions `switch` incluent un cas default.

  Le cas default doit toujours figurer en dernier dans l'instruction `switch`. Le cas `default` inclut une instruction `break` pour prévenir les erreurs fall-through en cas d'ajout d'un autre cas.
- Si un cas ne comporte pas d'instruction break, le cas comportera une erreur *fall through* (consultez `case A` dans l'exemple de code suivant).

  Votre instruction doit inclure un commentaire à la place de l'instruction `break`, comme l'illustre l'exemple suivant d'après `case A`. Dans cet exemple, si la condition correspond au cas A, les deux cas A et B s'exécutent.

Rédigez les instructions `switch` au format suivant :

```
switch (condition) {
case A :
  // instructions
  // fall-through
case B :
  // instructions
  break;
case Z :
  // instructions
  break;
default :
  // instructions
  break;
}
```

Écriture des instructions try..catch et try..catch..finally

Rédigez les instructions `try..catch` et `try..catch..finally` à l'aide du format suivant :

```
var myErr:Error;
// try..catch
try {
  // instructions
} catch (myErr) {
  // instructions
}

// try..catch..finally
try {
  // instructions
} catch (myErr) {
  // instructions
} finally {
  // instructions
}
```

Syntaxe des écouteurs

Vous pouvez écrire des écouteurs d'événements de plusieurs façons dans Flash 8 et ses versions ultérieures. Les techniques les plus populaires sont présentées dans les exemples suivants. Le premier exemple présente la syntaxe correcte, qui utilise un composant Loader pour charger du contenu dans un fichier SWF. L'événement `progress` commence avec le chargement du contenu et l'événement `complete` indique la fin de ce chargement.

```
var boxLdr:mx.controls.Loader;
var ldrListener:Object = new Object();
ldrListener.progress = function(evt:Object) {
  trace("loader loading:" + Math.round(evt.target.percentLoaded) + "%");
};
ldrListener.complete = function(evt:Object) {
  trace("loader complete:" + evt.target._name);
};
boxLdr.addEventListener("progress", ldrListener);
boxLdr.addEventListener("complete", ldrListener);
boxLdr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

Une légère variation du premier exemple de cette section consiste à utiliser la méthode `handleEvent`, mais cette technique est un peu plus lourde Adobe ne recommande pas cette technique dans la mesure où vous devez utiliser une série d'instructions `if..else` ou une instruction `switch` pour détecter l'événement à repérer.

```
var boxLdr:mx.controls.Loader;
var ldrListener:Object = new Object();

ldrListener.handleEvent = function(evt:Object) {
  switch (evt.type) {
  case "progress" :
    trace("loader loading:" + Math.round(evt.target.percentLoaded) + "%");
    break;
  case "complete" :
    trace("loader complete:" + evt.target._name);
    break;
  }
};
boxLdr.addEventListener("progress", ldrListener);
boxLdr.addEventListener("complete", ldrListener);
boxLdr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

ANNEXE A

Messages d'erreur

A

Adobe CS3 Professional fournit des fonctions de signalisation d'erreur de compilation si vous publiez dans ActionScript 2.0 (valeur par défaut). Le tableau suivant contient la liste des messages d'erreur que le compilateur Flash est susceptible de générer :

Numéro de l'erreur	Texte du message
1093	Un nom de classe était attendu.
1094	Un nom de classe de base est attendu après le mot-clé 'extends'.
1095	Utilisation incorrecte d'un attribut de membre.
1096	Impossible d'utiliser un nom de membre plusieurs fois.
1097	Toutes les fonctions de membre doivent avoir des noms.
1099	Cette instruction est interdite dans une définition de classe.
1100	Une classe ou une interface de ce nom est déjà définie.
1101	Incompatibilité de types.
1102	Il n'existe aucune classe nommée '‹ClassName›'.
1103	Il n'existe aucune propriété nommée '‹propertyName›'.
1104	Un appel de fonction pour un élément autre qu'une fonction a été tenté.
1105	Incompatibilité de types dans l'instruction d'affectation : [lhs-type] détecté au lieu de [rhs-type].
1106	Le membre est privé : accès impossible.
1107	Déclarations de variables interdites dans les interfaces.
1108	Déclarations d'événements interdites dans les interfaces.
1109	Déclarations de lecture/définitions interdites dans les interfaces.
1110	Membres privés interdits dans les interfaces.
1111	Corps de fonctions interdits dans les interfaces.

Numéro de l'erreur	Texte du message
1112	Une classe ne peut pas s'étendre.
1113	Une interface ne peut pas s'étendre.
1114	Aucune interface de ce nom n'est définie.
1115	Une classe ne peut pas étendre une interface.
1116	Une interface ne peut pas étendre une classe.
1117	Un nom d'interface est attendu après le mot-clé 'implements'.
1118	Une classe ne peut pas implémenter une autre classe, mais uniquement des interfaces.
1119	La classe doit implémenter la méthode 'methodName' depuis l'interface 'interfaceName'.
1120	La mise en œuvre d'une méthode d'interface doit être une méthode et non une propriété.
1121	Une classe ne peut pas étendre la même interface plusieurs fois.
1122	La mise en œuvre de la méthode d'interface ne correspond pas à sa définition.
1123	Cet élément est disponible uniquement dans ActionScript 1.0.
1124	Cet élément est disponible uniquement dans ActionScript 2.0.
1125	Membres statiques interdits dans les interfaces.
1126	L'expression renvoyée doit correspondre au type de renvoi de la fonction.
1127	Une instruction RETURN est requise dans cette fonction.
1128	Attribut utilisé en dehors de la classe.
1129	Une fonction dont le type de renvoi est Void ne renvoie aucune valeur.
1130	La clause 'extends' doit précéder la clause 'implements'.
1131	Un identifiant de type est attendu après ':'.
1132	Les interfaces doivent utiliser le mot-clé 'extends', et non pas 'implements'.
1133	Une classe ne peut pas étendre plus d'une classe.
1134	Une interface ne peut pas étendre plus d'une interface.
1135	La méthode nommée '‹methodName›' n'existe pas.
1136	Instruction interdite dans une définition d'interface.
1137	Une fonction set requiert un seul paramètre.
1138	Une fonction get ne requiert aucun paramètre.

Numéro de l'erreur	Texte du message
1139	Les classes ne peuvent être définies que dans des scripts de classe ActionScript 2.0 externes.
1140	Les scripts de classe ActionScript 2.0 peuvent définir uniquement des éléments de classe ou d'interface.
1141	Le nom de cette classe, '<A.B.C>', entre en conflit avec le nom d'une autre classe chargée, '<A.B>'. (Cette erreur se produit lorsque le compilateur ActionScript 2.0 ne peut pas compiler une classe parce que le nom complet d'une classe existante fait partie du nom de la classe créant un conflit. Par exemple, la compilation de la classe `mx.com.util` génère l'erreur 1141 si la classe `mx.com` est une classe compilée.)
1142	Impossible de charger la classe '<Nom de la classe ou de l'interface>'.
1143	Les interfaces ne peuvent être définies que dans des scripts de classe ActionScript 2.0 externes.
1144	Il est impossible d'accéder aux variables d'occurrence dans des fonctions statiques.
1145	Impossible d'imbriquer les définitions de classe et d'interface.
1146	La propriété référencée ne contient aucun attribut statique.
1147	L'appel de l'opérateur de super-classe ne correspond pas au superconstructeur.
1148	Seul l'attribut public est autorisé pour les méthodes d'interface.
1149	Impossible d'utiliser le mot-clé import en tant que directive.
1150	Vous devez exporter votre animation Flash au format Flash 7 pour utiliser cette action.
1151	Vous devez exporter votre animation Flash au format Flash 7 pour utiliser cette expression.
1152	Cette clause d'exception n'est pas placée correctement.
1153	Une classe doit comporter un seul constructeur.
1154	Un constructeur ne peut pas renvoyer de valeur.
1155	Un constructeur ne peut pas spécifier de type de renvoi.
1156	Une variable ne peut pas être de type Void.
1157	Un paramètre de fonction ne peut pas être de type Void.
1158	Accès aux membres statiques uniquement via les classes.

Numéro de l'erreur	Texte du message
1159	Plusieurs interfaces mises en œuvre contiennent la même méthode avec des types différents.
1160	Une classe ou une interface de ce nom existe déjà.
1161	Impossible de supprimer les classes, les interfaces et les types intégrés.
1162	Il n'existe aucune classe de ce nom.
1163	Le mot-clé '<keyword>' est réservé à ActionScript 2.0 et ne peut pas être utilisé ici.
1164	La définition de l'attribut personnalisé n'est pas terminée.
1165	Une seule classe ou interface peut être définie par fichier .as ActionScript 2.0.
1166	La classe en cours de compilation, '<A.b>', ne correspond pas à la classe importée, '<A.B>'. (Cette erreur se produit lorsqu'un nom de classe utilise une différente combinaison de majuscules par rapport à la classe importée. Par exemple, la compilation de la classe `mx.com.util` génère l'erreur 1166 si l'instruction `import mx.Com` figure dans le fichier util.as.)
1167	Vous devez indiquer un nom de classe.
1168	Le nom de classe que vous avez entré présente une erreur de syntaxe.
1169	Le nom d'interface que vous avez entré présente une erreur de syntaxe.
1170	Le nom de classe de base que vous avez entré présente une erreur de syntaxe.
1171	Le nom d'interface de base que vous avez entré présente une erreur de syntaxe.
1172	Vous devez indiquer un nom d'interface.
1173	Vous devez indiquer un nom de classe ou d'interface.
1174	Le nom de classe ou d'interface que vous avez entré présente une erreur de syntaxe.
1175	'variable' n'est pas accessible dans ce domaine.
1176	Plusieurs occurrences de l'attribut 'get/set/private/public/static' ont été trouvées.
1177	Un attribut de classe a été utilisé de manière incorrecte.
1178	Les variables et les fonctions d'occurrence ne peuvent pas être utilisées pour l'initialisation de variables statiques.

Numéro de l'erreur	Texte du message
1179	Des circularités d'exécution ont été détectées entre les classes suivantes : ‹liste de classes définies par l'utilisateur›. Cette erreur d'exécution indique que les classes personnalisées se référencent de façon incorrecte.
1180	La version Flash Player ciblée ne prend pas en charge le débogage.
1181	La version Flash Player ciblée ne prend pas en charge l'événement releaseOutside.
1182	La version Flash Player ciblée ne prend pas en charge l'événement dragOver.
1183	La version Flash Player ciblée ne prend pas en charge l'événement dragOut.
1184	La version Flash Player ciblée ne prend pas en charge les actions de déplacement.
1185	La version Flash Player ciblée ne prend pas en charge l'action loadMovie.
1186	La version Flash Player ciblée ne prend pas en charge l'action getURL.
1187	La version Flash Player ciblée ne prend pas en charge l'action FSCommand.
1188	Les instructions d'importation ne sont pas autorisées dans les définitions de classe ou d'interface.
1189	La classe '‹A.B›' ne peut pas être importée car son nom de feuille est déjà en cours de résolution en une classe en cours de définition, '‹C.B›'. (Par exemple, la compilation de la classe `util` génère l'erreur 1189 si l'instruction `import mx.util` figure dans le fichier util.as.)
1190	La classe '‹A.B›' ne peut pas être importée car son nom de feuille est déjà résolue en une classe importée précédemment '‹C.B›'. (Par exemple, la compilation de la classe `import jv.util` génère l'erreur 1190 si l'instruction `import mx.util` figure dans le fichier AS.)
1191	Les variables d'occurrence d'une classe peuvent être initialisées uniquement avec des expressions constantes au moment de la compilation.
1192	Les fonctions des membres de classe ne peuvent pas porter le même nom qu'une fonction constructeur de super-classe.
1193	Le nom de cette classe, '‹ClassName›', est en conflit avec le nom d'une autre classe chargée.
1194	Le corps d'un constructeur doit d'abord contenir un appel au superconstructeur.
1195	L'identifiant '‹className›' n'est pas résolu en un objet intégré '‹ClassName›' à l'exécution.

Numéro de l'erreur	Texte du message
1196	La classe '‹A.B.ClassName›' doit être définie dans un fichier dont le chemin d'accès relatif est ‹'A.B›'.
1197	Le caractère générique '*' est utilisé de manière incorrecte dans le nom de classe '‹ClassName›'.
1198	La fonction de membre '‹classname›' n'a pas la même casse que le nom de la classe en cours de définition, '‹ClassName›', et ne sera pas traitée en tant que constructeur de classe à l'exécution.
1199	Le seul type d'itérateur de boucle for-in accepté est le type Chaîne.
1200	Une fonction de définition ne peut pas renvoyer de valeur.
1201	Les seuls attributs autorisés pour les fonctions constructeur sont public et private.
1202	Impossible de trouver le fichier '‹toplevel.as›' requis pour la vérification du type de données ActionScript 2.0. Assurez-vous que le répertoire '‹$(LocalData)/Classes›' figure dans le chemin de classe global des préférences d'ActionScript.
1203	La branche entre ‹spanStart› et ‹spanEnd» dépasse une plage de 32 K.
1204	Impossible de trouver une classe ou un package portant le nom '‹packageName›' dans le package '‹PackageName›'.
1205	La version Flash Player ciblée ne prend pas en charge l'action FSCommand2.
1206	La fonction de membre '‹functionName›' est supérieure à 32 K.
1207	La fonction anonyme autour de la ligne ‹lineNumber› dépasse une plage de 32 K.
1208	Le code autour de la ligne ‹lineNumber› dépasse une plage de 32 K.
1210	Impossible d'utiliser également le nom du package '‹PackageName›' comme nom de méthode.
1211	Impossible d'utiliser également le nom du package '‹PackageName›' comme nom de propriété.
1212	Impossible de créer le fichier ASO pour la classe '‹ClassName›'. Assurez-vous que le nom de classe pleinement qualifié est suffisamment court pour que le nom de fichier ASO, '‹ClassName.aso›', comprennent moins de 255 caractères.
1213	Ce type de guillemet n'est pas autorisé dans ActionScript. Remplacez-les par des guillemets doubles standard (droits).

ANNEXE B

Opérateurs Flash 4 déconseillés

B

Le tableau suivant présente la liste des opérateurs Flash 4, qui sont déconseillés dans ActionScript2.0. Ne les utilisez pas, sauf si vous publiez vers Flash Player4 ou une version plus ancienne.

Opérateur	Description	Associativité
`not`	NON logique	Droite à gauche
`and`	AND logique	Gauche à droite
`or`	OU logique (Flash 4)	Gauche à droite
`add`	Concaténation de chaîne (auparavant &)	Gauche à droite
`lt`	Inférieur à (version chaîne)	Gauche à droite
`le`	Inférieur ou égal à (version chaîne)	Gauche à droite
`gt`	Supérieur à (version chaîne)	Gauche à droite
`ge`	Supérieur ou égal à (version chaîne)	Gauche à droite
`eq`	Egal (version chaîne)	Gauche à droite
`ne`	Différent (version chaîne)	Gauche à droite

ANNEXE C

Touches du clavier et valeurs de code correspondantes

C

Les tableaux suivants répertorient toutes les touches d'un clavier standard et les valeurs de code ASCII correspondantes qui permettent de les identifier dans ActionScript :

Vous pouvez utiliser les constantes de touche pour intercepter le comportement intégré des pressions de touche. Pour plus d'informations sur le gestionnaire on(), consultez la section gestionnaire on du *Guide de référence du langage ActionScript 2.0*. Pour capturer les valeurs de code et les valeurs de code ASCII à l'aide d'un fichier SWF en appuyant sur des touches, vous pouvez utiliser le code ActionScript suivant :

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  trace("DOWN -> Code: " + Key.getCode() + "\tACSII: " + Key.getAscii() +
  "\tKey: " + chr(Key.getAscii()));
};
Key.addListener(keyListener);
```

Pour plus d'informations sur la classe Key, consultez la section Key du *Guide de référence du langage ActionScript 2.0*. Pour intercepter des touches en testant un fichier SWF dans l'environnement auteur (Contrôle > Tester l'animation), veillez à sélectionner Contrôle > Désactiver les raccourcis clavier.

Lettres A à Z et chiffres (clavier standard) de 0 à 9

Le tableau suivant répertorie toutes les touches d'un clavier standard pour les lettres de A à Z et les chiffres de 0 à 9, avec les valeurs de code correspondantes utilisées pour identifier les touches dans ActionScript :

Touche alphabétique ou numérique	Code de touche	Code de touche ASCII
A	65	65
B	66	66
C	67	67
D	68	68
E	69	69
F	70	70
G	71	71
H	72	72
I	73	73
J	74	74
K	75	75
L	76	76
M	77	77
N	78	78
O	79	79
P	80	80
Q	81	81
R	82	82
S	83	83
T	84	84
U	85	85
V	86	86
W	87	87
X	88	88
Y	89	89
Z	90	90

Touche alphabétique ou numérique	Code de touche	Code de touche ASCII
0	48	48
1	49	49
2	50	50
3	51	51
4	52	52
5	53	53
6	54	54
7	55	55
8	56	56
9	57	57
a	65	97
b	66	98
c	67	99
d	68	100
e	69	101
f	70	102
g	71	103
h	72	104
i	73	105
j	74	106
k	75	107
l	76	108
m	77	109
n	78	110
o	79	111
p	80	112
q	81	113
r	82	114
s	83	115
t	84	116

Touche alphabétique ou numérique	Code de touche	Code de touche ASCII
u	85	117
v	86	118
w	87	119
x	88	120
y	89	121
z	90	122

Touches du clavier numérique

Le tableau suivant répertorie les touches d'un clavier numérique et indique les valeurs de code ASCII correspondantes permettant d'identifier les touches dans ActionScript :

Touche du clavier numérique	Code de touche	Code de touche ASCII
0 du pavé numérique	96	48
1 du pavé numérique	97	49
2 du pavé numérique	98	50
3 du pavé numérique	99	51
4 du pavé numérique	100	52
5 du pavé numérique	101	53
6 du pavé numérique	102	54
7 du pavé numérique	103	55
8 du pavé numérique	104	56
9 du pavé numérique	105	57
Multiplier	106	42
Additionner	107	43
Entrée	13	13
Soustraire	109	45
Décimal	110	46
Diviser	111	47

Touches de fonction

Le tableau suivant répertorie les touches de fonction d'un clavier standard et indique les valeurs de code correspondantes permettant d'identifier les touches dans ActionScript :

Touche de fonction	Code de touche	Code de touche ASCII
F1	112	0
F2	113	0
F3	114	0
F4	115	0
F5	116	0
F6	117	0
F7	118	0
F8	119	0
F9	120	0
F10	Cette touche est réservée par le système et ne peut pas être utilisée dans ActionScript.	Cette touche est réservée par le système et ne peut pas être utilisée dans ActionScript.
F11	122	0
F12	123	0
F13	124	0
F14	125	0
F15	126	0

Autres touches

Le tableau suivant répertorie les touches d'un clavier standard autres que les lettres, les chiffres, les touches de fonction et les touches du clavier numérique et indique les valeurs de code correspondantes permettant d'identifier les touches dans ActionScript :

Touche	Code de touche	Code de touche ASCII
Retour arrière	8	8
Tab	9	9
Entrée	13	13
Maj	16	0
Contrôle	17	0

Touche	Code de touche	Code de touche ASCII
Verr Maj	20	0
Echap	27	27
Espace	32	32
Pg. Préc.	33	0
Pg. Suiv.	34	0
Fin	35	0
Origine	36	0
Flèche gauche	37	0
Flèche vers le haut	38	0
Flèche droite	39	0
Flèche vers le bas	40	0
Insertion	45	0
Suppr	46	127
Verr num	144	0
Arrêt défil	145	0
Pause/Attn	19	0
; :	186	59
= +	187	61
- _	189	45
/ ?	191	47
` ~	192	96
[{	219	91
\ |	220	92
] }	221	93
" '	222	39
,	188	44
.	190	46
/	191	47

Pour connaître les codes et les valeurs ASCII d'autres touches, utilisez le code ActionScript situé au début de cette annexe et appuyez sur la touche désirée pour générer son code.

ANNEXE D

Ecriture de scripts destinés à des versions antérieures de Flash Player

D

ActionScript a considérablement évolué avec chaque version de l'outil de programmation Adobe Flash et Flash Player. Lorsque vous créez du contenu pour Adobe Flash Player 8, vous pouvez utiliser toutes les capacités d'ActionScript. Bien qu'il soit toujours possible de créer dans Flash du contenu destiné aux versions antérieures de Flash Player, vous ne pourrez pas exploiter tous les éléments d'ActionScript.

Ce chapitre donne des conseils pour écrire des scripts corrects du point de vue syntaxique pour la version de Flash Player ciblée.

REMARQUE

Vous pouvez consulter les études de pénétration de votre version de Flash Player sur le site Web d'Adobe ; consultez la page www.adobe.com/software/player_census/flashplayer/.

A propos du ciblage des versions antérieures de Flash Player

Lorsque vous écrivez vos scripts, utilisez les informations de disponibilité des éléments du *Guide de référence du langage ActionScript 2.0* pour déterminer si l'élément que vous voulez utiliser est pris en charge par la version de Flash Player que vous ciblez. Pour identifier les éléments disponibles, vous pouvez également afficher la boîte à outils Actions. Les éléments non pris en charge par votre version cible y sont surlignés en jaune.

Si vous créez du contenu pour Flash Player 6 et ses versions ultérieures, choisissez de préférence ActionScript 2.0, qui propose un certain nombre de fonctions importantes qui ne sont pas disponibles dans ActionScript 1.0, notamment de meilleures fonctions de signalisation des erreurs de compilation et des capacités de programmation orientée objet plus robustes.

Pour indiquer la version de Flash Player et d'ActionScript à utiliser lors de la publication d'un document, choisissez Fichier > Paramètres de publication, puis activez les options appropriées dans l'onglet Flash. Si vous visez Flash Player 4, consultez la section suivante.

Utilisation de Flash pour créer du contenu destiné à Flash Player 4

Pour utiliser Flash pour créer du contenu destiné à Flash Player 4, activez l'option Flash Player 4 dans l'onglet Flash de la boîte de dialogue Paramètres de publication (Fichier > Paramètres de publication).

Le langage ActionScript de Flash Player 4 ne possède qu'un seul type de données de base, qui est utilisé pour les manipulations de chaînes et de nombres. Lorsque vous développez une application pour Flash Player 4, vous devez utiliser les opérateurs de chaîne déconseillés indiqués dans la catégorie Eléments déconseillés > Opérateurs de la boîte à outils ActionScript.

Lorsque vous effectuez une publication pour Flash Player 4, vous pouvez utiliser les fonctions suivantes de Flash :

- L'opérateur d'accès tableau et objet (`[]`).
- L'opérateur point (`.`).
- Les opérateurs logiques, d'affectation, de pré-incrémentation et de post-incrémentation/décrémentation.
- L'opérateur modulo (`%`) et toutes les méthodes et propriétés de la classe Math.

Les éléments de langage suivants ne sont pas pris en charge de façon native par Flash Player 4. Flash les exporte sous la forme d'une série d'approximations, qui crée des résultats moins précis numériquement. De plus, du fait de l'ajout de séries d'approximations dans le fichier SWF, ces éléments de langage occupent plus d'espace dans les fichiers SWF de Flash Player 4 qu'ils n'en occupent dans Flash Player 5 ou dans les fichiers SWF d'une version ultérieure.

- Les actions `for`, `while`, `do..while`, `break` et `continue`.
- Les actions `print()` et `printAsBitmap()`.
- L'action `switch`.

Pour plus d'informations, consultez la section « A propos du ciblage des versions antérieures de Flash Player », à la page 773.

Utilisation de Flash pour ouvrir des fichiers Flash 4

Le code ActionScript de Flash 4 ne possédait qu'un seul véritable type de données : les chaînes. Il utilisait différents types d'opérateurs dans les expressions pour indiquer si la valeur devait être traitée comme une chaîne ou comme un nombre. Dans les versions ultérieures de Flash, vous pouvez utiliser un ensemble d'opérateurs sur tous les types de données.

Si vous utilisez la version 5 (ou ultérieure) de Flash pour ouvrir un fichier créé dans Flash 4, Flash convertit automatiquement les expressions ActionScript afin de les rendre compatibles avec la nouvelle syntaxe. Flash crée le type de donnée et effectue les conversions d'opérateur suivantes :

- L'opérateur = de Flash 4 était utilisé pour des égalités numériques. Dans la version 5 (et les versions ultérieures) de Flash, == est l'opérateur d'égalité et = est l'opérateur d'affectation. Tous les opérateurs = des fichiers Flash 4 sont automatiquement convertis en ==.
- Flash effectue automatiquement les conversions pour assurer le bon fonctionnement des opérateurs. L'introduction de plusieurs types de données donne une nouvelle signification aux opérateurs suivants :

  +, ==, !=, <>, <, >, >=, <=

  Dans ActionScript de Flash 4, ces opérateurs étaient toujours des opérateurs numériques. Dans la version 5 (et ultérieures) de Flash, ils se comportent différemment, selon le type de données des opérandes. Pour éviter les différences sémantiques dans les fichiers importés, la fonction `Number()` est insérée autour des opérandes de ces opérateurs. (Les nombres constants sont déjà des nombres évidents et ne sont donc pas inclus dans `Number()`). Pour plus d'informations sur ces opérateurs, consultez la table des opérateurs dans les sections « Priorité et associativité des opérateurs », à la page 149 et « Opérateurs Flash 4 déconseillés », à la page 765.
- Dans Flash 4, la séquence d'échappement `\n` générait le caractère de retour chariot (ASCII 13). A partir de la version 5 de Flash, conformément à la norme ECMA-262, `\n` génère le caractère de changement de ligne (ASCII 10). Une séquence `\n` dans les fichiers FLA de Flash 4 est automatiquement convertie en `\r`.
- L'opérateur `&` de Flash 4 était utilisé pour les additions de chaînes. A partir de la version 5, `&` devient l'opérateur AND au niveau du bit. L'opérateur d'addition de chaînes est maintenant appelé `add`. Tous les opérateurs `&` des fichiers de Flash 4 sont automatiquement convertis en opérateurs `add`.
- De nombreuses fonctions de Flash 4 ne nécessitaient pas l'usage de parenthèses de fermeture (par exemple, `Get Timer`, `Set Variable`, `Stop` et `Play`). Pour créer une syntaxe cohérente avec Flash 5 et les versions suivantes, la fonction `getTimer` et toutes les actions nécessitent maintenant l'usage de parenthèses de fermeture [`()`]. Ces parenthèses sont automatiquement ajoutées lors de la conversion.

- A partir de Flash 5, lors de l'exécution de la fonction `getProperty` pour un clip qui n'existe pas, la valeur `undefined` est renvoyée et non pas `0`. L'instruction `undefined == 0` a la valeur `false` dans ActionScript au delà de Flash 4 (dans Flash 4, `undefined == 1`). Dans Flash 5, pour résoudre ce problème lors de la conversion de fichiers Flash 4, insérez la fonction `Number()` dans les comparaisons d'égalité. Dans l'exemple suivant, `Number()` oblige à la conversion de `undefined` en `0` pour permettre la comparaison :

```
getProperty("clip", _width) == 0
Number(getProperty("clip", _width)) == Number(0)
```

REMARQUE

Si vous utilisez des mots-clés de Flash 5 ou d'une version ultérieure en tant que noms de variable dans votre code ActionScript Flash 4, la syntaxe renvoie une erreur lors de sa compilation dans Flash. Pour résoudre ce problème, renommez toutes vos variables. Pour plus d'informations, consultez les sections « Présentation des mots réservés », à la page 109 et « A propos de l'appellation des variables », à la page 57.

Utilisation de la syntaxe à barre oblique

La syntaxe à barre oblique (/) était utilisée dans Flash 3 et 4 pour indiquer le chemin cible d'un clip ou d'une variable. Avec la syntaxe à barre oblique, les barres obliques remplacent les points et les variables sont précédées par deux points, comme indiqué dans l'exemple suivant :

```
myMovieClip/childMovieClip:myVariable
```

Pour rédiger le même chemin cible en syntaxe à point, qui est également prise en charge dans Flash Player 5 (et versions ultérieures), utilisez la syntaxe suivante :

```
myMovieClip.childMovieClip.myVariable
```

La syntaxe à barre oblique était le plus souvent utilisée avec l'action `tellTarget` dont l'utilisation n'est plus recommandée. L'action `with` est maintenant privilégiée pour des raisons de compatibilité avec la syntaxe à point. Pour plus d'informations, reportez-vous à la `fonction tellTarget` et à l'`instruction with` dans le *Guide de référence du langage ActionScript 2.0*.

ANNEXE E

Programmation orientée objet avec ActionScript 1.0

E

Les informations contenues dans cette annexe sont extraites de la documentation d'Adobe Flash et concernent l'utilisation du modèle d'objet ActionScript 1.0 pour l'écriture de scripts. Elles sont ci-incluses pour les raisons suivantes :

- L'écriture de scripts orientés objet prenant en charge Flash Player 5 requiert l'utilisation d'ActionScript 1.0.
- Si vous utilisez déjà ActionScript 1.0 pour l'écriture de scripts orientés objet et ne souhaitez pas passer à ActionScript 2.0 dans l'immédiat, cette annexe vous permettra de trouver les informations nécessaires à l'écriture de vos scripts.

Si vous n'avez jamais utilisé ActionScript pour écrire des scripts orientés objet et que vos applications ne sont pas destinées à Flash Player 5, ne tenez pas compte des informations contenues dans cette annexe, car l'écriture de scripts orientés objet à l'aide d'ActionScript 1.0 n'est pas conseillée. Pour plus d'informations sur ActionScript 2.0, consultez le Chapitre 6, « Classes », à la page 197.

Ce chapitre contient les sections suivantes :

REMARQUE

Certains exemples de cette annexe utilisent la méthode Object.registerClass(). Cette dernière n'est prise en charge que par Flash Player 6 et les versions ultérieures ; veuillez ne pas l'utiliser pour Flash Player 5.

A propos d'ActionScript 1.0

REMARQUE

La plupart des utilisateurs de Flash ont tout intérêt à utiliser ActionScript 2.0, notamment pour les applications complexes. Pour plus d'informations sur ActionScript 2.0, consultez le Chapitre 6, « Classes », à la page 197.

ActionScript est un langage de programmation orienté objet. La programmation orientée objet utilise des *objets*, ou structures de données, afin de regrouper les propriétés et les méthodes qui contrôlent le comportement ou l'apparence de l'objet. Les objets permettent d'organiser et de réutiliser le code. Après avoir défini un objet, il est possible de se référer à ce dernier grâce à son nom, sans avoir à le redéfinir à chaque utilisation.

Une *classe* est une catégorie générique d'objets. Une classe définit une série d'objets ayant des propriétés communes et pouvant être contrôlés de la même manière. Les propriétés sont des attributs définissant un objet, tels que la taille, la position, la couleur, la transparence, etc. Les propriétés sont définies par classe ; les valeurs des propriétés sont définies pour chaque objet d'une classe. Les méthodes sont des fonctions pouvant définir ou rechercher les propriétés d'un objet. Par exemple, vous pouvez définir une méthode permettant de calculer la taille d'un objet. Tout comme les propriétés, les méthodes sont définies pour chaque classe d'objets, puis appelées pour chaque objet d'une classe.

ActionScript inclut plusieurs classes intégrées, comme la classe MovieClip, la classe Sound, etc. Vous pouvez également créer des classes personnalisées permettant de définir les catégories des objets de vos applications.

Les objets d'ActionScript peuvent être de simples conteneurs de données ou peuvent être représentés graphiquement sur la scène sous forme de clips, boutons ou champs de texte. Tous les clips sont des occurrences de la classe intégrée MovieClip, et tous les boutons sont des occurrences de la classe intégrée Button. Chaque occurrence d'un clip contient toutes les propriétés (par exemple, `_height`, `_rotation`, `_totalframes`) et toutes les méthodes (par exemple, `gotoAndPlay()`, `loadMovie()`, `startDrag()`) de la classe MovieClip.

Pour définir une classe, vous créez une fonction spéciale appelée *fonction constructeur*. Notez que les classes intégrées ont des fonctions constructeur intégrées. Par exemple, si vous souhaitez inclure dans votre application des informations sur un cycliste, vous pouvez créer une fonction constructeur, `Biker()`, avec les propriétés `time` et `distance` et la méthode `getSpeed()`, qui vous indique la vitesse à laquelle se déplace le cycliste :

```
function Biker(t, d) {
  this.time = t;
  this.distance = d;
  this.getSpeed = function() {return this.time / this.distance;};
}
```

Dans cet exemple, vous allez créer une fonction dont l'exécution requiert deux éléments d'information, ou *paramètres* : `t` et `d`. Lorsque vous appelez la fonction pour créer de nouvelles occurrences de l'objet, vous lui transférez les paramètres. Le code suivant crée des occurrences de l'objet Biker appelées `emma` et `hamish`, puis suit la vitesse de l'instance `emma`, en utilisant la méthode `getSpeed()` du code ActionScript précédent :

```
emma = new Biker(30, 5);
hamish = new Biker(40, 5);
trace(emma.getSpeed()); // affiche 6 dans le panneau de sortie
```

Dans la création de scripts orientés objet, les classes peuvent échanger entre elles des propriétés et des méthodes selon un ordre spécifique, ce qui est appelé *héritage*. Vous pouvez utiliser l'héritage pour développer ou redéfinir les propriétés et les méthodes d'une classe. Une classe qui hérite d'une autre classe est appelée *sous-classe*. Une classe qui transmet des propriétés et des méthodes à une autre classe est appelée *super-classe*. Une classe peut être à la fois une sous-classe et une super-classe.

Un objet est un type de données complexe contenant aucune ou plusieurs propriétés et méthodes. Chaque propriété, tout comme une variable, possède un nom et une valeur. Les propriétés sont liées à l'objet et contiennent des valeurs qui peuvent être modifiées ou récupérées. Ces valeurs peuvent être de n'importe quel type de données : chaîne, nombre, booléen, objet, clip ou undefined (non défini). Les propriétés suivantes sont de différents types de données :

```
customer.name = "Jane Doe";
customer.age = 30;
customer.member = true;
customer.account.currentRecord = 609;
customer.mcInstanceName._visible = true;
```

La propriété d'un objet peut également être un objet. A la ligne 4 de l'exemple précédent, `account` est une propriété de l'objet `customer` et `currentRecord` est une propriété de l'objet `account`. La propriété `currentRecord` est de type nombre.

Création d'un objet personnalisé dans ActionScript 1.0

REMARQUE

La plupart des utilisateurs de Flash ont tout intérêt à utiliser ActionScript 2.0, notamment pour les applications complexes. Pour plus d'informations sur ActionScript 2.0, consultez le Chapitre 6, « Classes », à la page 197.

Pour créer un objet personnalisé, définissez une fonction constructeur. Une fonction constructeur reçoit toujours le nom du type d'objet qu'elle sert à créer. Vous pouvez utiliser le mot-clé `this` dans le corps de la fonction constructeur, afin de faire référence à l'objet créé par le constructeur (lorsque vous appelez une fonction constructeur, Flash transmet `this` à la fonction en tant que paramètre masqué). Dans l'exemple suivant, la fonction constructeur crée un cercle avec la propriété `radius` :

```
function Circle(radius) {
  this.radius = radius;
}
```

Après avoir défini la fonction constructeur, vous devez créer une occurrence de l'objet. Utilisez l'opérateur `new` devant le nom de la fonction constructeur, puis donnez un nom de variable à la nouvelle occurrence. Par exemple, le code suivant utilise l'opérateur `new` pour créer un objet Cercle d'un rayon de 5 et l'affecte à la variable `myCircle` :

```
myCircle = new Circle(5);
```

REMARQUE

Un objet est du même domaine que la variable à laquelle il est affecté.

Affectation de méthodes à un objet personnalisé dans ActionScript 1.0

REMARQUE

La plupart des utilisateurs de Flash ont tout intérêt à utiliser ActionScript 2.0, notamment pour les applications complexes. Pour plus d'informations sur ActionScript 2.0, consultez le Chapitre 6, « Classes », à la page 197.

Vous pouvez définir les méthodes d'un objet à l'intérieur de la fonction constructeur de l'objet. Cependant, cette technique n'est pas recommandée dans la mesure où elle définit la méthode à chaque fois que vous utilisez la fonction constructeur. L'exemple suivant crée les méthodes `getArea()` et `getDiameter()` : et suit la zone et le diamètre de l'occurrence créée, `myCircle`, avec un rayon de 55 :

```
function Circle(radius) {
  this.radius = radius;
  this.getArea = function(){
    return Math.PI * this.radius * this.radius;
  };
  this.getDiameter = function() {
    return 2 * this.radius;
  };
}
var myCircle = new Circle(55);
trace(myCircle.getArea());
trace(myCircle.getDiameter());
```

Chaque fonction constructeur a une propriété `prototype` qui est créée automatiquement en même temps que la fonction. La propriété `prototype` indique les valeurs de propriétés par défaut des objets créés avec cette fonction. Chaque nouvelle occurrence d'un objet possède une propriété `__proto__`, qui fait référence à la propriété `prototype` de la fonction constructeur qui a servi à la créer. Ainsi, si vous affectez des méthodes à une propriété `prototype` d'un objet, ces dernières sont disponibles pour toute occurrence nouvellement créée de cet objet. Mieux vaut affecter une méthode à la propriété `prototype` de la fonction constructeur, car elle existe à un endroit et est référencée par de nouvelles occurrences de l'objet (ou classe). Vous pouvez utiliser les propriétés `prototype` et `__proto__` pour étendre des objets, ce qui permet de réutiliser du code selon une méthode orientée objet. (Pour plus d'informations, voir « Création d'héritages dans ActionScript 1.0 », à la page 785.)

La procédure suivante explique comment affecter une méthode `getArea()` à un objet Cercle personnalisé.

Pour affecter une méthode à un objet personnalisé :

1. Définissez la fonction constructeur `Circle()`, comme suit :

```
function Circle(radius) {
  this.radius = radius;
}
```

2. Définissez la méthode `getArea()` de l'objet Cercle. La méthode `getArea()` calcule l'aire du cercle. Dans l'exemple suivant, vous pouvez utiliser un littéral de fonction pour définir la méthode `getArea()` et affecter la propriété `getArea` à l'objet prototype du cercle :

```
Circle.prototype.getArea = function () {
    return Math.PI * this.radius * this.radius;
};
```

3. L'exemple suivant crée une occurrence de l'objet Cercle :

```
var myCircle = new Circle(4);
```

4. Appelez la méthode `getArea()` du nouvel objet `myCircle` avec le code suivant :

```
var myCircleArea = myCircle.getArea();
trace(myCircleArea); // affiche 50.265 dans le panneau Sortie...
```

   ActionScript cherche l'objet `myCircle` pour la méthode `getArea()`. Puisque l'objet n'a pas de méthode `getArea()`, son objet prototype `Circle.prototype` est recherché pour la méthode `getArea()`. ActionScript le recherche, l'appelle et suit `myCircleArea`.

Définition des méthodes du gestionnaire d'événement dans ActionScript 1.0

REMARQUE

La plupart des utilisateurs de Flash ont tout intérêt à utiliser ActionScript 2.0, notamment pour les applications complexes. Pour plus d'informations sur ActionScript 2.0, consultez le Chapitre 6, « Classes », à la page 197.

Vous pouvez créer une classe ActionScript pour les clips et définir les méthodes du gestionnaire d'événement dans l'objet prototype de cette nouvelle classe. La définition des méthodes dans l'objet prototype permet à toutes les occurrences de ce symbole de répondre de la même manière à ces événements.

Vous pouvez également ajouter une méthode de gestionnaire d'événement `onClipEvent()` ou `on()` à une instance individuelle afin de fournir des instructions uniques qui ne s'exécutent que lorsque l'événement se produit pour cette occurrence. Les méthodes `onClipEvent()` et `on()` ne supplantent pas la méthode du gestionnaire d'événement ; ainsi, les deux événements exécutent leurs scripts. Toutefois, si vous définissez les méthodes de gestionnaire d'événement dans l'objet prototype et que vous définissez également une méthode de gestionnaire d'événement pour une occurrence précise, la définition de l'occurrence supplante celle du prototype.

Pour définir une méthode de gestionnaire d'événement dans un objet prototype d'un objet :

1. Créez un symbole de clip et définissez son ID de liaison sur `theID` en sélectionnant le symbole dans le panneau Bibliothèque, puis en sélectionnant Liaison dans le menu contextuel Bibliothèque.
2. Dans le panneau Actions (Fenêtre > Actions), utilisez l'instruction `function` pour définir une nouvelle classe, comme indiqué ici :

   ```
   // définir une classe
   function myClipClass() {}
   ```

   Cette nouvelle classe sera affectée à toutes les occurrences du clip ajoutées à l'application par le scénario, ou qui y seront ajoutées via les méthodes `attachMovie()` ou `duplicateMovieClip()`. Si vous souhaitez que ces clips accèdent aux méthodes et propriétés de l'objet intégré MovieClip, vous devez faire en sorte que la nouvelle classe hérite de la classe MovieClip.
3. Entrez du code, comme dans l'exemple suivant :

   ```
   // hériter de la classe MovieClip
   myClipClass.prototype = new MovieClip();
   ```

   La classe `myClipClass` héritera désormais de toutes les propriétés et méthodes de la classe MovieClip.
4. Entrez un code semblable au suivant pour définir les méthodes de gestionnaire d'événement de la nouvelle classe :

   ```
   // définir des méthodes de gestionnaire d'événement pour la
   // classe myClipClass
   myClipClass.prototype.onLoad = function() {trace("movie clip loaded");}
   myClipClass.prototype.onEnterFrame = function() {trace("movie clip
     entered frame");}
   ```
5. Choisissez Fenêtre > Bibliothèque pour ouvrir le panneau Bibliothèque, s'il n'est pas déjà visible.
6. Sélectionnez les symboles à associer à la nouvelle classe, puis sélectionnez Liaison dans le menu contextuel du panneau Bibliothèque.
7. Dans la boîte de dialogue Propriétés de liaison, activez l'option Exporter pour ActionScript.
8. Entrez un identifiant de liaison dans la zone de texte Identifiant.

   L'identifiant de liaison doit être identique pour tous les symboles que vous voulez associer à la nouvelle classe. Dans l'exemple `myClipClass`, l'identifiant est `theID`.

9. Entrez du code tel que le suivant dans le panneau Actions :

```
// enregistrer la classe
Object.registerClass("theID", myClipClass);
this.attachMovie("theID","myName",1);
```

Cela permet d'enregistrer le symbole dont l'identifiant du lien est `theID` et dont la classe est `myClipClass`. Toutes les occurrences de `myClipClass` ont des méthodes de gestionnaire d'événement se comportant comme définies à l'étape 4. Elles se comportent également comme toutes les occurrences de la classe MovieClip, puisque vous avez fait en sorte que la nouvelle classe hérite de la classe MovieClip dans l'étape 3.

Le code complet est présenté dans l'exemple suivant :

```
function myClipClass(){}

myClipClass.prototype = new MovieClip();
myClipClass.prototype.onLoad = function(){
  trace("movie clip loaded");
}
myClipClass.prototype.onPress = function(){
  trace("pressed");
}

myClipClass.prototype.onEnterFrame = function(){
  trace("movie clip entered frame");
}

myClipClass.prototype.myfunction = function(){
  trace("myfunction called");
}

Object.registerClass("myclipID",myClipClass);
this.attachMovie("myclipID","clipName",3);
```

Création d'héritages dans ActionScript 1.0

REMARQUE

La plupart des utilisateurs de Flash ont tout intérêt à utiliser ActionScript 2.0, notamment pour les applications complexes. Pour plus d'informations sur ActionScript 2.0, consultez le Chapitre 6, « Classes », à la page 197.

L'héritage est un moyen d'organiser, d'étendre et de réutiliser une fonctionnalité. Les sous-classes héritent des propriétés et des méthodes des super-classes, auxquelles elles ajoutent leurs propres propriétés et méthodes spécialisées. Par exemple, dans le monde réel, Cycle pourrait être une super-classe dont VTT et Tricycle seraient des sous-classes. Ces deux sous-classes contiennent ou *héritent* des méthodes et propriétés de la super-classe (par exemple, `roues`). Chaque sous-classe possède également des propriétés et des méthodes spécifiques qui étendent la super-classe (par exemple, la sous-classe VTT pourrait comporter une propriété `pignons`). Vous pouvez utiliser les éléments `prototype` et `__proto__` pour créer un héritage dans ActionScript.

Toutes les fonctions constructeur ont une propriété `prototype` qui est créée automatiquement en même temps que la fonction. La propriété `prototype` indique les valeurs de propriétés par défaut des objets créés avec cette fonction. Vous pouvez utiliser la propriété `prototype` pour affecter des propriétés et méthodes à une classe. (Pour plus d'informations, voir « Affectation de méthodes à un objet personnalisé dans ActionScript 1.0 », à la page 781.)

Toutes les occurrences d'une classe possèdent une propriété `__proto__` qui indique de quel objet elles sont héritières. Lorsque vous utilisez une fonction constructeur pour créer un objet, la propriété `__proto__` est configurée de manière à faire référence à la propriété `prototype` de sa fonction constructeur.

L'héritage fonctionne selon une hiérarchie précise. Quand vous appelez une propriété ou une méthode d'un objet, ActionScript examine l'objet pour vérifier si cet élément existe. Si tel n'est pas le cas, ActionScript recherche les informations (`myObject.__proto__`) dans la propriété `__proto__` de l'objet. Si la propriété n'est pas une propriété de l'objet `__proto__` de l'objet, ActionScript recherche alors dans `myObject.__proto__.__proto__` et ainsi de suite.

L'exemple suivant définit la fonction constructeur `Bike()` :

```
function Bike(length, color) {
  this.length = length;
  this.color = color;
  this.pos = 0;
}
```

Le code suivant ajoute la méthode `roll()` à la classe Bike :

```
Bike.prototype.roll = function() {return this.pos += 20;};
```

Vous pouvez ensuite suivre la position du vélo avec le code ci-dessous :

```
var myBike = new Bike(55, "blue");
trace(myBike.roll());   // affiche 20 dans le panneau de sortie.
trace(myBike.roll());   // affiche 40 dans le panneau de sortie.
```

Au lieu d'ajouter une méthode `roll()` aux classes VTT et Tricycle, vous pouvez créer la classe VTT et définir Cycle en tant que super-classe, comme indiqué dans l'exemple suivant :

```
MountainBike.prototype = new Bike();
```

Vous devez ensuite appeler la méthode `roll()` de VTT, comme ci-dessous :

```
var myKona = new MountainBike(20, "teal");
trace(myKona.roll());   // affiche 20 dans le panneau de sortie
```

Les clips n'héritent pas les uns des autres. Pour créer un héritage entre clips, vous pouvez utiliser la méthode `Object.registerClass()` afin d'affecter une classe autre que MovieClip à des clips.

Ajout de propriétés de lecture/définition à des objets dans ActionScript 1.0

REMARQUE

La plupart des utilisateurs de Flash ont tout intérêt à utiliser ActionScript 2.0, notamment pour les applications complexes. Pour plus d'informations sur ActionScript 2.0, consultez le Chapitre 6, « Classes », à la page 197.

Vous pouvez créer des propriétés de lecture/définition d'un objet à l'aide de la méthode `Object.addProperty()`.

Une *fonction de lecture* est une fonction sans paramètre. La valeur renvoyée peut être de n'importe quel type. Son type peut changer d'une invocation à l'autre. La valeur renvoyée est considérée comme la valeur actuelle de la propriété.

Une *fonction de définition* est une fonction qui prend un paramètre, qui correspond à la nouvelle valeur de la propriété. Par exemple, si la propriété x est affectée par l'instruction x = 1, la fonction de définition transmise est le paramètre 1 du numéro de type. La valeur renvoyée par la fonction de définition est ignorée.

Lorsque Flash lit une propriété de lecture/définition, il appelle la fonction get et la valeur renvoyée devient une valeur de `prop`. Lorsque Flash écrit une propriété de lecture/définition, il ouvre la fonction de définition et transmet la nouvelle valeur comme paramètre. Si une propriété portant le nom donné existe déjà, la nouvelle propriété la remplace.

Vous pouvez ajouter des propriété de lecture/définition à des objets prototypes. Dans ce cas, toutes les occurrences d'objet qui héritent de l'objet prototype héritent de la propriété de lecture/définition. Vous pouvez ajouter une propriété de lecture/définition à un endroit, au niveau de l'objet prototype, et de la propager à toutes les occurrences d'une classe, tout comme lorsque vous ajoutez des méthodes à des objets prototypes. Si une fonction de lecture/définition est invoquée pour une propriété de lecture/définition dans un objet prototype hérité, la référence transmise à la fonction de lecture/définition sera l'objet originellement référencé et non l'objet prototype.

La commande Déboguer > Lister les variables en mode test prend en charge les propriétés de lecture/définition ajoutés aux objets à l'aide de la méthode `Object.addProperty()`. Les propriétés ainsi ajoutées à un objet s'affichent avec les autres propriétés de l'objet dans le panneau de sortie. Les propriétés de lecture/définition sont identifiées dans le panneau de sortie par le préfixe `[lecture/définition]`. Pour plus d'informations sur la commande des variables de liste, consultez le guide *Utilisation de Flash*.

Utilisation des propriétés de l'objet Function dans ActionScript 1.0

REMARQUE

La plupart des utilisateurs de Flash ont tout intérêt à utiliser ActionScript 2.0, notamment pour les applications complexes. Pour plus d'informations sur ActionScript 2.0, consultez le Chapitre 6, « Classes », à la page 197.

Vous pouvez spécifier l'objet auquel une fonction est appliquée, ainsi que les valeurs de paramètre transmises à cette fonction à l'aide des méthodes `call()` et `apply()` de l'objet Function. Toutes les fonctions dans ActionScript sont représentées par un objet Function, de sorte que toutes les fonctions prennent en charge les méthodes `call()` et `apply()`. Lorsque vous créez une classe personnalisée à l'aide de la fonction constructeur ou lorsque vous définissez les méthodes d'une classe personnalisée à l'aide d'une fonction, vous pouvez ouvrir les méthodes `call()` et `apply()` de la fonction.

Invocation d'une fonction à l'aide de la méthode Function.call dans ActionScript 1.0

REMARQUE

La plupart des utilisateurs de Flash ont tout intérêt à utiliser ActionScript 2.0, notamment pour les applications complexes. Pour plus d'informations sur ActionScript 2.0, consultez le Chapitre 6, « Classes », à la page 197.

La méthode `Function.call()` ouvre la fonction représentée par un objet Function.

Dans presque tous les cas, l'opérateur d'appel de fonction (`()`) peut être utilisé au lieu de la méthode `call()`. L'opérateur de la fonction call crée un code concis et lisible. La méthode `call()` est surtout utile lorsque le paramètre `this` de l'invocation de fonction doit être explicitement contrôlé. Normalement, si une fonction est invoquée en tant que méthode d'un objet, dans le corps de la fonction, `this` est défini sur `myObject` comme suit :

```
myObject.myMethod(1, 2, 3);
```

Dans certains cas, vous aurez besoin que `this` pointe autre part, si par exemple une fonction doit être invoquée comme méthode d'objet alors qu'elle n'est pas stockée en tant que méthode de cet objet, comme indiqué dans l'exemple suivant :

```
myObject.myMethod.call(myOtherObject, 1, 2, 3);
```

Vous pouvez transmettre la valeur `null` pour le paramètre *`thisObject`* pour invoquer une fonction en tant que fonction ordinaire et pas en tant que méthode d'un objet. Par exemple, les invocations de fonction suivantes sont équivalentes :

```
Math.sin(Math.PI / 4)
Math.sin.call(null, Math.PI / 4)
```

Pour invoquer une fonction à l'aide de la méthode Function.call() :

- Utilisez la syntaxe suivante :

  `myFunction``.call(`*`thisObject, parameter1, ..., parameterN`*`)`

  Cette méthode prend en charge les paramètres suivants :

  - Le paramètre *`thisObject`* spécifie la valeur `this` dans le corps de la fonction.
  - Les paramètres *`parameter1...`*, *`parameterN`* spécifient les paramètres devant être transmis à *`myFunction`*. Vous pouvez spécifier zéro ou plusieurs paramètres.

Spécification de l'objet auquel une fonction est appliquée à l'aide de Function.apply() dans ActionScript 1.0

REMARQUE

La plupart des utilisateurs de Flash ont tout intérêt à utiliser ActionScript 2.0, notamment pour les applications complexes. Pour plus d'informations sur ActionScript 2.0, consultez le Chapitre 6, « Classes », à la page 197.

La méthode `Function.apply()` spécifie la valeur `this` à utiliser dans toute fonction appelée par ActionScript. Cette méthode spécifie également les paramètres à transmettre à toute fonction appelée.

Les paramètres sont spécifiés sous forme d'objet Array. Ceci est souvent utile lorsque le nombre de paramètres à transmettre n'est pas connu avant l'exécution du script.

Pour plus d'informations, consultez la section `apply (méthode Function.apply)` du *Guide de référence du langage ActionScript 2.0*.

Pour spécifier l'objet auquel une fonction est appliquée à l'aide de Function.apply() :

- Utilisez la syntaxe suivante :

  myFunction`.apply(`*thisObject*`, `*argumentsObject*`)`

  Cette méthode prend en charge les paramètres suivants :

  - Le paramètre *thisObject* spécifie l'objet auquel *myFunction* s'applique.
  - Le paramètre *argumentsObject* définit un tableau dont les éléments sont transmis dans *myFunction* en tant que paramètres.

Index

Symboles

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X