MX

macromedia®
FLASH™MX
2004

Utilisation des composants

Remerciements

Directeur : Erick Vera

Gestion de projet : Stephanie Gowin, Barbara Nelson

Rédaction : Jody Bleyle, Mary Burger, Kim Diezel, Stephanie Gowin, Dan Harris, Barbara Herbert, Barbara Nelson, Shirley Ong, Tim Statler

Rédactrice en chef : Rosana Francescato

Révision : Mary Ferguson, Mary Kraemer, Noreen Maher, Antonio Padial, Lisa Stanziano, Anne Szabla

Gestion de la production : Patrice O'Neill

Conception du support et production : Adam Barnett, Christopher Basmajian, Aaron Begley, John Francis, Jeff Harmon

Localisation: Tim Hussey, Seungmin Lee, Masayo Noda, Simone Pux, Yuko Yagi, Florian de Joannès

Première édition : Août 2003

Macromedia, Inc.
600 Townsend St.
San Francisco, CA 94103
Etats-Unis

TABLE DES MATIERES

INTRODUCTION
Présentation des composants

Macromedia Flash MX 2004 et Flash MX Professionnel 2004 sont les outils standard des professionnels pour la création de contenu web percutant. La création de ces applications Internet riches repose sur des unités élémentaires appelées composants. Un composant est un clip qui contient des paramètres définis pendant la phase de programmation dans Macromedia Flash, ainsi que des API ActionScript permettant de personnaliser le composant lors de l'exécution. Les composants sont conçus pour permettre aux développeurs de réutiliser et de partager du code. Ils permettent également d'encapsuler une fonctionnalité complexe que les concepteurs peuvent utiliser et personnaliser sans avoir à se servir d'ActionScript.

Les composants sont basés sur la version 2 (v2) de l'architecture des composants Macromedia. Celle-ci permet de créer facilement et rapidement des applications robustes à la présentation et au comportement cohérents. Ce manuel explique comment créer des applications avec les composants v2 et présente les API (interface de programmation) des composants. Il inclut des scénarios d'utilisation et des exemples de procédures à suivre pour utiliser les composants v2 de Flash MX 2004 ou Flash MX Professionnel 2004, par ordre alphabétique, ainsi qu'une description des API des composants.

Vous pouvez utiliser les composants créés par Macromedia, télécharger des composants créés par d'autres développeurs ou créer vos propres composants.

Public visé

Ce manuel est destiné aux développeurs qui créent des applications Flash MX 2004 ou Flash MX Professionnel 2004 et qui souhaitent utiliser des composants pour accélérer le développement. Vous devez déjà avoir des notions du développement des applications dans Macromedia Flash, de la rédaction d'instructions ActionScript et de Macromedia Flash Player.

Ce manuel présume que vous avez déjà installé Flash MX 2004 ou Flash MX Professionnel 2004 et que vous savez comment l'utiliser. Avant d'utiliser les composants, nous vous conseillons d'étudier la leçon « Créer une interface utilisateur avec des composants » (sélectionnez Aide > Comment > Manuel de prise en main rapide > Créer une interface utilisateur avec des composants).

Si vous souhaitez rédiger le moins d'instructions ActionScript possible, vous pouvez faire glisser les composants dans un document, définir leurs paramètres dans l'inspecteur des propriétés ou dans le panneau Inspecteur de composants et associer un gestionnaire `on()` directement au composant dans le panneau Actions afin de gérer les événements de composants.

Si vous êtes programmeur et que vous souhaitez créer des applications plus robustes, vous pouvez créer les composants dynamiquement, utiliser leurs API pour définir les propriétés et appeler les méthodes à l'exécution. Vous pouvez également utiliser le modèle d'événement écouteur pour gérer les événements.

Pour plus d'informations, consultez le Chapitre 2, *Utilisation des composants*, page 15.

Configuration système requise

Aucune configuration particulière n'est requise pour les composants Macromedia outre Flash MX 2004 ou Flash MX Professionnel 2004.

Installation des composants

Un jeu de composants Macromedia est déjà installé lorsque vous lancez Flash MX 2004 ou Flash MX Professionnel 2004 pour la première fois. Vous pouvez les visualiser dans le panneau Composants.

Flash MX 2004 inclut les composants suivants :

- *Composant Button*
- *Composant CheckBox*
- *Composant ComboBox*
- *Composant Label*
- *Composant List*
- *Composant Loader*
- *Composant NumericStepper*
- *Classe PopUpManager*
- *Composant ProgressBar*
- *Composant RadioButton*
- *Composant ScrollPane*
- *Composant TextArea*
- *Composant TextInput*
- *Composant Window*

Flash MX Professionnel 2004 inclut les composants de Flash MX 2004, plus les classes et composants suivants :

- *Composant Accordion*
- *Composant Alert*
- *Paquet DataBinding*
- *Composant DateField*
- *Composant DataGrid*
- *Composant DataHolder*
- *Composant DataSet*
- *Composant DateChooser*
- *Classe Form*

- *Composant MediaController*
- *Composant MediaDisplay*
- *Composant MediaPlayback*
- *Composant Menu*
- *Composant RDBMSResolver*
- *Classe Screen*
- *Composant Tree*
- *Composant WebServiceConnector*
- *Composant XMLConnector*
- *Composant XUpdateResolver*

Pour vérifier l'installation des composants Flash MX 2004 ou Flash MX Professionnel 2004 :

1 Démarrez Flash.

2 Choisissez Fenêtre > Panneaux de développement > Composants pour ouvrir le panneau Composants s'il n'est pas déjà ouvert.

3 Choisissez UI Components pour développer l'arborescence et afficher les composants installés.

Vous pouvez également télécharger des composants à partir de Macromedia Exchange. Pour installer des composants téléchargés à partir d'Exchange, téléchargez et installez Macromedia Extension Manager.

Tous les composants, qu'il s'agisse de fichiers SWC ou FLA (consultez *A propos des clips compilés et des fichiers SWC*, page 14), peuvent apparaître dans le panneau Composants de Flash. Suivez les étapes suivantes pour installer les composants sur un ordinateur Windows ou Macintosh.

Pour installer des composants sur un ordinateur Windows ou Macintosh :

1 Quittez Flash.

2 Placez le fichier SWC ou FLA contenant le composant dans le dossier suivant, sur votre disque dur :

- DD/Applications/Macromedia Flash MX 2004/First Run/Components (Macintosh)
- \Program Files\Macromedia\Flash MX 2004\<language>\First Run\Components (Windows)

3 Ouvrez Flash.

4 Choisissez Fenêtre > Panneaux de développement > Composants pour visualiser le composant dans le panneau Composants s'il n'est pas déjà ouvert.

A propos de la documentation

Ce document explique comment utiliser les composants pour développer des applications Flash. Il présume que le lecteur connaît déjà Macromedia Flash et ActionScript. Une documentation spécifique est disponible séparément sur Flash et les produits associés.

- Pour plus d'informations sur Macromedia Flash, consultez *Utilisation de Flash*, le *Guide de référence ActionScript* ainsi que l'aide du Dictionnaire ActionScript.
- Pour plus d'informations sur l'accès aux services web avec des applications Flash, consultez *Using Flash Remoting*.

Conventions typographiques

Ce manuel utilise les conventions typographiques suivantes :

- *La police en italique* indique une valeur qui devrait être remplacée (par exemple, dans le chemin d'un dossier).
- `La police de code` indique le code ActionScript.
- *`La police de code en italique`* identifie les paramètres ActionScript.
- **La police en gras** indique une entrée que vous devez saisir exactement à l'identique.

  Remarque : La police en gras est différente de la police utilisée pour les en-têtes répétés. La police utilisée pour les en-têtes répétés constitue une alternative aux puces.

Termes employés dans ce manuel

Ce manuel utilise les termes suivants :

à l'exécution Lorsque le code est exécuté dans Flash Player.

pendant la programmation Lors de l'utilisation de l'environnement auteur de Flash.

Ressources supplémentaires

Pour les informations les plus récentes sur Flash, ainsi que des conseils d'utilisateurs experts, des rubriques avancées, des exemples, des conseils et autres mises à jour, consultez le site web de Macromedia DevNet, régulièrement mis à jour. Consultez régulièrement ce site web pour vous tenir au courant des nouveautés de Flash et tirer le meilleur parti de votre programme.

Pour des TechNotes, des mises à jour de la documentation et des liens vers des ressources supplémentaires dans la communauté Flash, consultez le Centre de support Macromedia Flash à l'adresse www.macromedia.com/support/flash.

Pour plus d'informations sur les termes, la syntaxe ActionScript et son utilisation, consultez le *Guide de référence ActionScript* et l'aide du Dictionnaire ActionScript.

CHAPITRE 1
A propos des composants

Les composants sont des clips qui contiennent des paramètres permettant d'en modifier l'aspect et le comportement. Un composant peut offrir n'importe quelle fonctionnalité imaginée par son créateur. Il peut s'agir d'un simple contrôle d'interface utilisateur, comme un bouton radio ou une case à cocher, ou d'un contenant, comme un panneau défilant. Un composant peut être invisible, comme le gestionnaire de focus (FocusManager) qui permet de contrôler quels objets reçoivent du focus dans une application.

Les composants permettent de créer des applications Macromedia Flash MX 2004 et Macromedia Flash MX Professionnel 2004 complexes, même sans bien connaître ActionScript. Plutôt que de créer des boutons personnalisés, des listes et listes déroulantes, vous pouvez faire glisser ces composants à partir du panneau Composants pour ajouter des fonctionnalités à vos applications. Vous pouvez également personnaliser facilement l'aspect des composants pour les adapter à vos besoins.

Les composants sont basés sur la version 2 (v2) de l'architecture des composants Macromedia. Celle-ci permet de créer facilement et rapidement des applications robustes à la présentation et au comportement cohérents. L'architecture v2 inclut des classes sur lesquelles sont basés tous les composants, des styles et des mécanismes d'enveloppe qui vous permettent de personnaliser l'aspect des composants, un modèle d'événement diffuseur/écouteur, la gestion de la profondeur et du focus, la mise en œuvre de l'accessibilité et bien d'autres choses encore.

Tous les composants contiennent des paramètres prédéfinis que vous pouvez configurer pendant la programmation dans Flash. Chaque composant a également un jeu unique de méthodes, de propriétés et d'événements ActionScript, également appelé *API* (interface de programmation), qui permet de définir les paramètres et d'autres options à l'exécution.

Flash MX 2004 et Flash MX Professionnel 2004 incluent de nombreux composants Flash inédits et plusieurs nouvelles versions de composants déjà inclus dans Flash MX. Pour une liste complète des composants inclus dans Flash MX 2004 et Flash MX Professionnel 2004, consultez *Installation des composants*, page 8. Vous pouvez également télécharger des composants développés par des membres de la communauté Flash dans Macromedia Exchange.

Avantages des composants v2

Les composants permettent de séparer le code de la conception. Ils permettent également de réutiliser du code, soit dans les composants que vous avez créés, soit en téléchargeant et en installant des composants créés par d'autres développeurs.

Les composants permettent aux programmeurs de créer des fonctionnalités que les concepteurs pourront utiliser dans les applications. Les développeurs peuvent encapsuler les fonctionnalités fréquemment utilisées dans des composants. Les concepteurs peuvent, quant à eux, personnaliser l'aspect et le comportement de ces composants en modifiant leurs paramètres dans l'inspecteur des propriétés ou dans le panneau Inspecteur de composants.

Les membres de la communauté Flash peuvent utiliser le site Macromedia Exchange pour échanger des composants. En utilisant des composants, vous n'avez plus besoin de concevoir chaque élément depuis le début pour créer une application web complexe. Vous pouvez trouver les composants requis et les placer dans un document Flash pour créer une nouvelle application.

Les composants basés sur l'architecture des composants v2 utilisent les mêmes fonctionnalités de base : styles, traitement des événements, application d'enveloppes, gestion du focus et de la profondeur. Lorsque vous ajoutez le premier composant v2 dans une application, environ 25 Ko sont ajoutés dans le document qui fournit ces fonctionnalités de base. Lorsque vous ajoutez des composants supplémentaires, ces mêmes 25 Ko sont réutilisés pour ces composants. La taille du document augmente donc moins que vous avez pu l'imaginer. Pour plus d'informations sur la mise à niveau des composants v1 en composants v2, consultez *Mise à niveau des composants v1 vers l'architecture v2*, page 26.

Catégories de composants

Les composants inclus dans Flash MX 2004 et Flash MX Professionnel 2004 sont divisés en quatre catégories : les composants de l'interface utilisateur (IU), les composants de support, les composants de données et les gestionnaires. Les contrôles d'interface utilisateur permettent à l'utilisateur d'interagir avec une application ; par exemple, les composants RadioButton, CheckBox et TextInput sont des contrôles d'interface utilisateur. Les composants de support permettent de lire un support dans une application : le composant MediaPlayback est un composant de support. Les composants de données permettent de charger et de manipuler des informations provenant de sources de données ; les composants WebServiceConnector et XMLConnector sont des composants de données. Les gestionnaires sont des composants invisibles qui permettent de gérer une fonction, telle que le focus ou la profondeur, dans une application ; les composants FocusManager, DepthManager, PopUpManager et StyleManager sont les composants gestionnaires inclus dans Flash MX 2004 et Flash MX Professionnel 2004. Pour une liste complète des composants de chaque catégorie, consultez le Chapitre 4, *Dictionnaire des composants*, page 47.

Architecture des composants

Vous pouvez utiliser l'inspecteur des propriétés ou le panneau Inspecteur de composants pour modifier les paramètres des composants afin d'en utiliser la fonctionnalité de base. Cependant, si vous voulez contrôler davantage la fonctionnalité des composants, vous devez utiliser leurs API et comprendre la façon dont ils ont été créés.

Flash MX 2004 et Flash MX Professionnel 2004Les composants sont basés sur la version 2 (v2) de l'architecture des composants Macromedia et Les composants de la version 2 sont supportés par Flash Player 6 et Flash Player 7. Ils ne sont pas toujours compatibles avec les composants basés sur la version 1 (v1) de l'architecture (tous les composants antérieurs à Flash MX 2004). Les composants v1 ne sont pas non plus supportés par Flash Player 7. Pour plus d'informations, consultez *Mise à niveau des composants v1 vers l'architecture v2*, page 26.

Les composants v2 sont inclus dans le panneau Composants en tant que symboles Clip compilé (SWC). Un clip compilé est un clip composant dont le code a été compilé. Les clips compilés contiennent des aperçus en direct intégrés et ne peuvent pas être modifiés, mais comme pour les autres composants, il est possible de changer leurs paramètres dans l'inspecteur des propriétés et dans le panneau Inspecteur de composants. Pour plus d'informations, consultez *A propos des clips compilés et des fichiers SWC*, page 14.

Les composants v2 sont rédigés en ActionScript 2. Chaque composant est une classe et chaque classe est un paquet ActionScript. Par exemple, un composant de bouton radio est une occurrence de la classe RadioButton dont le nom de paquet est `mx.controls`. Pour plus d'informations sur les paquets, consultez « Utilisation de paquets », dans le Guide de référence ActionScript de l'aide.

Tous les composants créés avec la version 2 de l'architecture des composants Macromedia sont des sous-classes des classes UIObject et UIComponent et héritent de tous les événements, propriétés et méthodes de ces classes. De nombreux composants sont également des sous-classes d'autres composants. Le chemin de l'héritage de chaque composant est indiqué à l'entrée correspondante du Chapitre 4, *Dictionnaire des composants*, page 47.

Tous les composants utilisent également le même modèle d'événement, les mêmes styles CSS et le même mécanisme intégré d'application d'enveloppes. Pour plus d'informations sur les styles et l'application des enveloppes, consultez le Chapitre 3, *Personnalisation des composants*, page 27. Pour plus d'informations sur le traitement des événements, consultez le Chapitre 2, *Utilisation des composants*, page 15.

Nouveautés des composants v2

Le panneau Inspecteur de composants permet de modifier les paramètres des composants pendant la programmation dans Macromedia Flash et Macromedia Dreamweaver (consultez *Composants dans le panneau Inspecteur de composants et dans l'inspecteur des propriétés*, page 16).

Le modèle d'événement écouteur permet aux objets d'écoute des fonctions de traiter les événements (consultez *A propos des événements de composant*, page 22).

Les propriétés des enveloppes permettent de charger des états uniquement lorsque vous en avez besoin (consultez *A propos de l'application des enveloppes aux composants*, page 37).

Les styles CSS permettent d'obtenir un aspect cohérent dans toutes les applications (consultez *Utilisation des styles pour personnaliser la couleur et le texte des composants*, page 27).

Les thèmes permettent d'appliquer un nouvel aspect à un jeu de composants (consultez *A propos des thèmes*, page 35).

Le thème Halo fournit une interface utilisateur prête à l'emploi, flexible et précise pour les applications.

Les classes Manager fournissent un moyen aisé de traiter le focus et le codage dans une application. Pour plus d'informations, consultez *Création de la navigation personnalisée du focus*, page 25 et *Gestion de la profondeur des composants dans un document*, page 25.

Les classes de base UIObject et UIComponent fournissent les fonctionnalités élémentaires de tous les composants. Pour plus d'informations, consultez *Classe UIComponent*, page 231 et *Classe UIObject*, page 240.

Le format de fichier SWC permet une distribution aisée et l'utilisation d'un code dissimulable. Consultez *Création de composants*. Il vous faudra parfois télécharger le fichier PDF le plus récent sur le site du Centre de support Flash pour obtenir ces informations.

La fonction intégrée de liaison des données est disponible dans le panneau Inspecteur de composants. Pour plus d'informations sur cette fonction, appuyez sur le bouton Mise à jour de l'aide.

La hiérarchie des classes facilement extensible avec ActionScript 2 permet de créer des espaces de nom uniques, d'importer des classes si nécessaire et d'établir des sous-classes afin d'étendre les composants. Consultez *Création de composants* et le *Guide de référence ActionScript*. Il vous faudra parfois télécharger le fichier PDF le plus récent sur le site du Centre de support Flash pour obtenir ces informations.

A propos des clips compilés et des fichiers SWC

Les clips compilés sont utilisés pour effectuer la compilation préalable des symboles complexes dans les documents Flash. Il est par exemple possible de transformer en clip compilé un clip contenant un volume important de code ActionScript qui ne change pas souvent. Cela entraîne un temps d'exécution inférieur pour les commandes Tester l'animation et Publier.

Le format de fichier SWC est utilisé pour enregistrer et distribuer les composants. Lorsque vous placez un fichier SWC dans le dossier First Run/Components, le composant apparaît dans le panneau Composants. Lorsque vous ajoutez un composant sur la scène à partir du panneau Composants, le symbole d'un clip compilé est ajouté dans la bibliothèque.

Pour plus d'informations sur les fichiers SWC, consultez *Création de composants*. Il vous faudra parfois télécharger le fichier PDF le plus récent sur le site du Centre de support Flash pour obtenir ces informations.

Accessibilité et composants

Le contenu publié sur le web est accessible à partir des quatre coins de la planète – il devrait donc l'être également pour les personnes souffrant de divers handicaps. Le contenu visuel des animations Flash peut être rendu accessible aux personnes souffrant de handicaps visuels à l'aide d'un logiciel adapté pour la lecture de description audio du contenu de l'écran.

Lors de la création des composants, l'auteur peut rédiger une instruction ActionScript destinée à activer la communication entre les composants et un lecteur d'écran. Ensuite, lorsqu'un développeur utilise ces composants pour créer une application dans Flash, il utilise le panneau Accessibilité pour configurer toutes les occurrences des composants.

La plupart des composants créés par Macromedia sont conçus pour être accessibles. Pour savoir si un composant est accessible, consultez son entrée dans le Chapitre 4, *Dictionnaire des composants*, page 47. Lorsque vous créez une application dans Flash, vous devez ajouter une ligne de code pour chaque composant (`mx.accessibility.`*`ComponentName`*`AccImpl.enableAccessibility();`) et définir ses paramètres d'accessibilité dans le panneau Accessibilité. L'accessibilité fonctionne de la même manière pour les composants que pour les autres clips Flash. Pour plus d'informations, consultez « Création de contenu accessible », dans le guide Utilisation de Flash de l'aide. Il vous faudra peut-être mettre le système d'aide à jour pour obtenir ces informations.

Vous pouvez également utiliser le clavier pour accéder à la plupart des composants créés par Macromedia. Les entrées respectives des composants du Chapitre 4, *Dictionnaire des composants*, page 47 indiquent si vous pouvez ou non les contrôler à l'aide du clavier.

CHAPITRE 2
Utilisation des composants

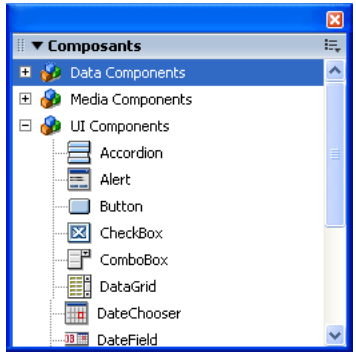
Il existe plusieurs manières d'utiliser les composants dans Macromedia Flash MX 2004 et Macromedia Flash MX Professionnel 2004. La visualisation des composants et leur ajout à un document au cours de la programmation s'effectuent dans le panneau Composants. Une fois qu'un composant a été ajouté à un document, vous pouvez visualiser ses propriétés dans l'inspecteur des propriétés ou dans le panneau Inspecteur de composants. Pour faire communiquer les composants entre eux, vous devez écouter leurs événements et les traiter avec ActionScript. Vous pouvez également gérer la profondeur du composant dans un document et contrôler le moment où il reçoit du focus.

Le panneau Composants

Tous les composants sont stockés dans le panneau Composants. Lorsque vous installez Flash MX 2004 ou Flash MX Professionnel 2004 et que vous le lancez pour la première fois, les composants du dossier Macromedia Flash 2004/First Run/Components (Macintosh) ou Macromedia\Flash 2004\<langue>\First Run\Components (Windows) s'affichent dans le panneau Composants.

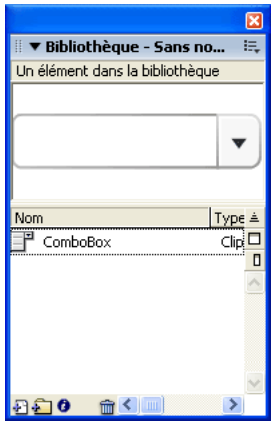
Pour afficher le panneau Composants :

- Choisissez Fenêtre > Panneaux de développement > Composants.



Composants dans le panneau Bibliothèque

Lorsque vous ajoutez un composant à un document, il apparaît sous la forme d'un symbole de clip compilé (SWC) dans le panneau Bibliothèque.



Composant ComboBox dans le panneau Bibliothèque.

Vous pouvez ajouter plusieurs occurrences d'un composant en faisant glisser son icône de la bibliothèque sur la scène.

Pour plus d'informations sur les clips compilés, consultez *Utilisation des fichiers SWC et des clips compilés*, page 18.

Composants dans le panneau Inspecteur de composants et dans l'inspecteur des propriétés

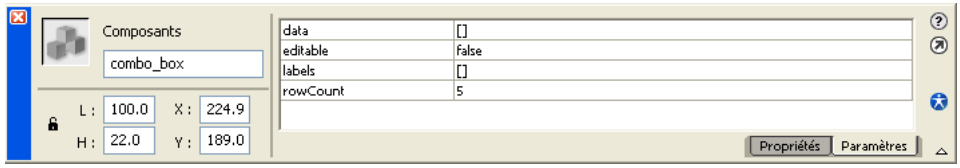
Après avoir ajouté une occurrence d'un composant dans un document Flash, utilisez l'inspecteur des propriétés pour définir et consulter des informations à son sujet. Les occurrences d'un composant sont créées par glisser-déposer sur la scène à partir du panneau Composants, puis en nommant l'occurrence dans l'inspecteur des propriétés et en définissant ses paramètres au moyen des champs de l'onglet Paramètres. Vous pouvez également définir les paramètres d'une occurrence de composant à l'aide du panneau Inspecteur de composants. Le panneau utilisé pour définir les paramètres importe peu ; c'est une question de préférence personnelle. Pour plus d'informations sur la définition des paramètres, consultez *Définition des paramètres des composants*, page 21.

Pour consulter des informations au sujet d'une occurrence de composant dans l'inspecteur des propriétés :

1 Choisissez Fenêtre > Propriétés.

2 Sélectionnez une occurrence de composant sur la scène.

3 Pour consulter les paramètres, cliquez sur l'onglet Paramètres.

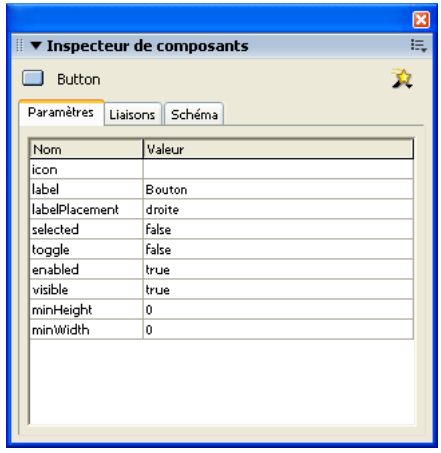


Pour consulter les paramètres d'une occurrence de composant dans le panneau Inspecteur de composants :

1 Choisissez Fenêtre > Inspecteur de composants.

2 Sélectionnez une occurrence de composant sur la scène.

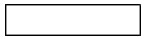
3 Pour consulter les paramètres, cliquez sur l'onglet Paramètres.



Composants dans l'aperçu en direct

La fonction Aperçu en direct, activée par défaut, permet de visualiser les composants sur la scène tels qu'ils apparaîtront dans le contenu Flash publié, et de connaître leur taille approximative. L'aperçu en direct reflète différents paramètres de différents composants. Pour plus d'informations sur les paramètres de composant qui apparaissent dans l'aperçu en direct, consultez les entrées des composants dans le Chapitre 4, *Dictionnaire des composants*, page 47. Les composants de l'aperçu en direct ne fonctionnent pas. Pour tester la fonctionnalité d'un composant, vous pouvez utiliser la commande Contrôle > Tester l'animation.

Composant Button avec Aperçu en direct activé.



Composant Button avec Aperçu en direct désactivé.

Pour activer ou désactiver l'aperçu en direct:

- Choisissez Contrôle > Activer l'aperçu en direct. Une coche en regard de l'option indique qu'elle est activée.

Pour plus d'informations, consultez *Création de composants*. Il vous faudra parfois télécharger le fichier PDF le plus récent sur le site du Centre de support Flash pour obtenir ces informations.

Utilisation des fichiers SWC et des clips compilés

Les composants inclus dans Flash MX 2004 ou Flash MX Professionnel 2004 ne sont pas des fichiers FLA ; ce sont des fichiers SWC. SWC est le format de fichier que Macromedia utilise pour les composants. Lorsque vous ajoutez un composant sur la scène à partir du panneau Composants, le symbole d'un clip compilé est ajouté dans la bibliothèque. Les fichiers SWC sont des clips compilés qui ont été exportés en vue d'être distribués.

Un clip peut également être « compilé » dans Flash et converti en symbole « Clip compilé ». Le symbole du clip compilé se comporte exactement comme le symbole du clip à partir duquel il a été compilé, mais les clips compilés sont affichés et publiés beaucoup plus vite que les symboles de clips normaux. Les clips compilés ne peuvent pas être modifiés, mais leurs propriétés apparaissent dans l'inspecteur des propriétés et dans le panneau Inspecteur de composants, et ils incluent un aperçu en direct.

Les composants inclus dans Flash MX 2004 ou Flash MX Professionnel 2004 ont déjà été transformés en clips compilés. Si vous créez un composant, vous pouvez choisir de l'exporter en tant que fichier SWC pour le distribuer. Pour plus d'informations, consultez *Création de composants*. Il vous faudra parfois télécharger le fichier PDF le plus récent sur le site du Centre de support Flash pour obtenir ces informations.

Pour compiler un symbole de clip :

- Choisissez le clip dans la bibliothèque et cliquez avec le bouton droit (Windows) ou maintenez la touche Contrôle enfoncée (Macintosh) tout en sélectionnant Convertir en clip compilé.

Pour exporter un fichier SWC :

- Choisissez le clip dans la bibliothèque et cliquez avec le bouton droit (Windows) ou maintenez la touche Contrôle enfoncée (Macintosh) tout en sélectionnant Exporter le fichier SWC.

Remarque : Flash MX 2004 et Flash MX Professionnel 2004 continuent de supporter les composants FLA.

Ajout de composants aux documents Flash

Lorsque vous faites glisser un composant pour le déplacer du panneau Composants vers la scène, un symbole de clip compilé est ajouté dans le panneau Bibliothèque. Une fois que le symbole de clip compilé se trouve dans la bibliothèque, vous pouvez également ajouter ce composant au document en cours d'exécution en utilisant la méthode `UIObject.createClassObject()` d'ActionScript.

- Les utilisateurs novices de Flash peuvent utiliser le panneau Composants pour ajouter des composants à un document Flash, définir des paramètres de base au moyen de l'inspecteur des propriétés ou du panneau Paramètres de composant, puis utiliser le gestionnaire d'événements `on()` pour contrôler les composants.

- Les utilisateurs intermédiaires de Flash peuvent utiliser le panneau Composants pour ajouter des composants à un document Flash, puis utiliser l'inspecteur des propriétés, les méthodes ActionScript ou une combinaison des deux pour définir les paramètres. Ils peuvent utiliser le gestionnaire d'événements `on()` ou des écouteurs d'événements pour traiter les événements de composants.
- Les programmeurs expérimentés peuvent utiliser une combinaison du panneau Composants et d'ActionScript pour ajouter des composants et définir des propriétés, ou décider d'utiliser uniquement ActionScript pour définir les occurrences de composants en cours d'exécution. Ils peuvent utiliser les écouteurs d'événements pour contrôler les composants.

Si vous modifiez les enveloppes d'un composant, puis que vous ajoutez une autre version de celui-ci ou d'un composant qui partage ces enveloppes, vous pouvez choisir d'utiliser les enveloppes modifiées, ou de les remplacer par un nouveau jeu d'enveloppes par défaut. Si vous remplacez les enveloppes modifiées, tous les composants qui les utilisent sont mis à jour afin d'utiliser leurs versions par défaut. Pour plus d'informations sur la manipulation des enveloppes, consultez le Chapitre 3, *Personnalisation des composants*, page 27.

Ajout de composants à l'aide du panneau Composants

Après avoir ajouté un composant à un document au moyen du panneau Composants, vous pouvez ajouter d'autres occurrences de ce composant au document en les faisant glisser du panneau Bibliothèque sur la scène. Vous pouvez définir les propriétés des occurrences supplémentaires dans l'onglet Paramètres de l'inspecteur des propriétés ou dans le panneau Paramètres de composant.

Pour ajouter un composant à un document Flash avec le panneau Composants :

1. Choisissez Fenêtre > Composants.
2. Procédez de l'une des manières suivantes :
   - Faites glisser un composant du panneau Composants jusqu'à la scène.
   - Double-cliquez sur un composant du panneau Composants.
3. Si le composant est un fichier FLA (tous les composants v2 installés sont des fichiers SWC) *et* si vous avez modifié des enveloppes pour une autre occurrence du même composant ou d'un composant qui partage des enveloppes avec celui que vous ajoutez, effectuez l'une des opérations suivantes :
   - Choisissez Ne pas remplacer les éléments existants pour conserver les enveloppes modifiées et les appliquer au nouveau composant.
   - Choisissez Remplacer les éléments existants pour remplacer toutes les enveloppes par les enveloppes par défaut. Le nouveau composant et toutes ses versions précédentes, ou les versions précédentes des composants qui partagent ses enveloppes, utiliseront les enveloppes par défaut.
4. Sélectionnez le composant sur la scène.
5. Choisissez Fenêtre > Propriétés.
6. Dans l'inspecteur des propriétés, entrez un nom pour l'occurrence de composant.
7. Cliquez sur l'onglet Paramètres et définissez les paramètres de l'occurrence.

   Pour plus d'informations, consultez *Définition des paramètres des composants*, page 21.
8. Modifiez la taille du composant.

Pour plus d'informations sur le dimensionnement des types spécifiques de composants, consultez les entrées des composants dans le Chapitre 4, *Dictionnaire des composants*, page 47.

9 Modifiez éventuellement la couleur et la mise en forme du texte du composant, en effectuant une ou plusieurs des opérations suivantes:

- Définissez ou modifiez la valeur d'une propriété de style spécifique pour une occurrence de composant au moyen de la méthode `setStyle()` disponible pour tous les composants. Pour plus d'informations, consultez *`UIObject.setStyle()`*.
- Modifiez les propriétés souhaitées sur la déclaration de style _global affectée à tous les composants v2.
- Si nécessaire, créez une déclaration de style personnalisée pour des occurrences de composant spécifiques.

  Pour plus d'informations, consultez *Utilisation des styles pour personnaliser la couleur et le texte des composants*, page 27.

10 Personnalisez éventuellement l'apparence du composant en effectuant l'une des opérations suivantes :

- Appliquer un thème (consultez *A propos des thèmes*, page 35).
- Modifier les enveloppes d'un composant (consultez *A propos de l'application des enveloppes aux composants*, page 37).

Ajout de composants à l'aide d'ActionScript

Pour ajouter un composant à un document à l'aide d'ActionScript, vous devez d'abord l'ajouter à la bibliothèque.

Vous pouvez utiliser les méthodes ActionScript afin de définir des paramètres supplémentaires pour les composants ajoutés dynamiquement. Pour plus d'informations, consultez le Chapitre 4, *Dictionnaire des composants*, page 47.

Remarque : Les instructions figurant dans cette section supposent que vous possédez une connaissance intermédiaire ou avancée d'ActionScript.

Pour ajouter un composant à votre document Flash avec ActionScript :

1 Faites glisser un composant du panneau Composants vers la scène et supprimez-le.

  Cette action permet d'ajouter le composant à la bibliothèque.

2 Sélectionnez l'image, dans le scénario, où vous souhaitez placer le composant.

3 Ouvrez le panneau Actions s'il n'est pas déjà ouvert.

4 Appelez la méthode `createClassObject()` pour créer l'occurrence du composant au moment de l'exécution.

  Cette méthode peut être appelée seule ou à partir de n'importe quelle occurrence de composant. Elle prend pour paramètres un nom de classe de composant, un nom d'occurrence pour la nouvelle occurrence, une profondeur et un objet d'initialisation facultatif. Vous pouvez spécifier le paquet de classes dans le paramètre *`className`* de la manière suivante :

```
createClassObject(mx.controls.CheckBox, "cb", 5, {label:"A cocher"});
```

  Vous pouvez importer le paquet de classes de la manière suivante :

```
import mx.controls.CheckBox;
createClassObject(CheckBox, "cb", 5, {label:"A cocher"});
```

  Pour plus d'informations, consultez *`UIObject.createClassObject()`*.

5 Utilisez les méthodes ActionScript et les propriétés du composant pour définir d'autres options ou remplacer les paramètres définis pendant sa création.

Pour des informations détaillées sur les méthodes ActionScript et les propriétés disponibles pour chaque composant, consultez leurs entrées respectives dans le Chapitre 4, *Dictionnaire des composants*, page 47.

A propos de la taille des étiquettes, de la largeur et de la hauteur des composants

Si une occurrence de composant ajoutée à un document n'est pas assez grande pour que l'étiquette soit affichée, le texte de l'étiquette est coupé. Si l'occurrence d'un composant est plus grande que le texte, la zone de cible dépassera les limites de l'étiquette.

Utilisez l'outil Transformation libre ou la méthode `setSize()` pour redimensionner les occurrences du composant. Vous pouvez appeler la méthode `setSize()` à partir de n'importe quelle occurrence de composant (consultez *`UIObject.setSize()`*). Si vous utilisez les propriétés ActionScript `_width` et `_height` pour régler la largeur et la hauteur d'un composant, le redimensionnement est effectué mais la disposition du contenu reste la même. Le composant risque alors d'être déformé pendant la lecture de l'animation. Pour plus d'informations sur le dimensionnement des composants, consultez leurs entrées respectives dans le Chapitre 4, *Dictionnaire des composants*, page 47.

Définition des paramètres des composants

Chaque composant a des paramètres que vous pouvez définir afin de modifier son aspect et son comportement. Un paramètre est une propriété ou une méthode qui apparaît dans l'inspecteur des propriétés et dans le panneau Inspecteur de composants. Les propriétés et méthodes les plus couramment utilisées apparaissent sous la forme de paramètres de création ; les autres doivent être définies à l'aide d'ActionScript. Tous les paramètres pouvant être définis en cours de programmation peuvent également être définis avec ActionScript. La définition d'un paramètre avec ActionScript remplace toutes les valeurs définies en cours de programmation.

Tous les composants v2 héritent des propriétés et des méthodes de la classe UIObject et de la classe UIComponent ; il s'agit des propriétés et des méthodes utilisées par tous les composants, telles que *`UIObject.setSize()`*, *`UIObject.setStyle()`*, *`UIObject.x`* et *`UIObject.y`*. Chaque composant a aussi des propriétés et des méthodes uniques, certaines d'entre elles étant disponibles en tant que paramètres de création. Par exemple, le composant ProgressBar a une propriété `percentComplete` (*`ProgressBar.percentComplete`*) et le composant NumericStepper a des propriétés `nextValue` et `previousValue` (*`NumericStepper.nextValue`*, *`NumericStepper.previousValue`*).

Suppression de composants des documents Flash

Pour supprimer les occurrences d'un composant dans un document Flash, vous devez supprimer le composant de la bibliothèque en effaçant l'icône du clip compilé.

Pour supprimer un composant d'un document :

1 Dans le panneau Bibliothèque, choisissez le symbole du clip compilé (SWC).

2 Cliquez sur le bouton Supprimer, en bas du panneau Bibliothèque, ou choisissez Supprimer dans le menu des options de la bibliothèque.

3 Dans la boîte de dialogue Supprimer, cliquez sur Supprimer pour confirmer la suppression.

Utilisation des conseils de code

Lorsque vous utilisez ActionScript 2, vous pouvez uniquement taper une variable basée sur une classe intégrée, une classe de composant par exemple. Dans ce cas, l'éditeur ActionScript affiche des conseils de code pour la variable. Par exemple, supposons que vous tapez ce qui suit :

```
var maCaseAcocher:CheckBox
maCaseAcocher.
```

Dès que vous saisissez le point, Flash affiche une liste des méthodes et des propriétés disponibles pour les composants CheckBox, puisque vous avez saisi la variable comme étant de type Case à cocher. Pour plus d'informations sur la saisie des données, consultez « Typage strict des données », dans le Guide de référence ActionScript de l'aide. Pour plus d'informations sur l'utilisation des conseils de code lors de leur apparition, consultez « Utilisation des conseils de code », dans le Guide de référence ActionScript de l'aide.

A propos des événements de composant

Tous les composants contiennent des événements qui sont diffusés lorsque l'utilisateur établit une interaction avec le composant ou s'il arrive quelque chose de significatif au composant. Pour traiter un événement, vous devez rédiger le code ActionScript exécuté lorsque l'événement est déclenché.

Vous pouvez traiter les événements de composant en appliquant les méthodes suivantes :

- Utilisez le gestionnaire d'événements de composant `on()`.
- Utilisez des écouteurs d'événements.

Utilisation des gestionnaires d'événements de composant

L'utilisation du gestionnaire d'événements de composant `on()` est le moyen le plus facile de traiter les événements de composant. Vous pouvez affecter le gestionnaire `on()` à une occurrence de composant, de la même manière que vous affectez un gestionnaire à un bouton ou à un clip.

Le mot-clé `this`, utilisé dans un gestionnaire `on()` joint à un composant, correspond à l'occurrence de comportement. Par exemple, le code suivant, joint à l'occurrence de composant Button `monComposantBouton`, envoie « _level0.monComposantBouton » au panneau de sortie :

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

Pour utiliser le gestionnaire on() :

1 Faites glisser un composant CheckBox vers la scène à partir du panneau Composants.

2 Sélectionnez le composant et choisissez Fenêtre > Actions.

3 Dans le panneau Actions, saisissez le code suivant :

```
on(click){
  trace("La case a été cochée");
}
```

Vous pouvez saisir le code de votre choix entre les accolades ({}).

4 Choisissez Contrôle > Tester l'animation et cliquez sur la case pour visualiser le tracé dans le panneau de sortie.

Pour plus d'informations, consultez les entrées respectives des événements dans le Chapitre 4, *Dictionnaire des composants*, page 47.

Utilisation des écouteurs d'événements de composant

L'utilisation des écouteurs est la manière la plus puissante de traiter les événements de composant. Les événements sont diffusés par les composants et tous les objets enregistrés sur le diffuseur (occurrence de composant) en tant qu'écouteurs peuvent être prévenus de l'événement. Une fonction traitant l'événement est affectée à l'écouteur. Vous pouvez enregistrer plusieurs écouteurs sur une occurrence de composant. Vous pouvez également enregistrer un écouteur sur plusieurs occurrences de composant.

Pour utiliser le modèle d'écouteur d'événements, vous devez créer un objet d'écoute avec une propriété correspondant au nom de l'événement. La propriété est affectée à une fonction de rappel. Vous appelez ensuite la méthode *`UIEventDispatcher.addEventListener()`* sur l'occurrence de composant qui diffuse l'événement et vous lui passez le nom de l'événement et le nom de l'objet d'écoute. On appelle l'appel de la méthode *`UIEventDispatcher.addEventListener()`* « enregistrement » ou « souscription » d'un écouteur, comme dans l'exemple qui suit :

```
objetDécoute.eventName = function(evtObj){
  // votre code ici
};
occurrenceDeComposant.addEventListener("eventName", objetDécoute);
```

Dans le code ci-dessus, le mot-clé `this`, s'il est utilisé dans la fonction de rappel, a un domaine limité à `objetDécoute`.

Le paramètre `evtObj` est un objet événement automatiquement généré lorsqu'un événement est déclenché et passé à la fonction de rappel de l'objet d'écoute. L'objet événement a des propriétés qui contiennent des informations sur l'événement. Pour plus d'informations, consultez *Classe UIEventDispatcher*, page 238.

Pour plus d'informations sur les événements diffusés par un composant, consultez les entrées respectives des composants dans le Chapitre 4, *Dictionnaire des composants*, page 47.

Pour enregistrer un écouteur d'événements, procédez comme suit :

1 Faites glisser un composant Button vers la scène à partir du panneau Composants.

2 Dans l'inspecteur des propriétés, saisissez le nom d'occurrence **button**.

3 Faites glisser un composant TextInput vers la scène à partir du panneau Composants.

4 Dans l'inspecteur des propriétés, saisissez le nom d'occurrence **monTexte**.

5 Sélectionnez l'image 1 dans le scénario.

6 Choisissez Fenêtre > Actions.

7 Dans le panneau Actions, saisissez le code suivant :

```
form = new Object();
form.click = function(evt){
  monTexte.text = evt.target;
}
button.addEventListener("click", form);
```

La propriété cible de l'objet événement est une référence à l'occurrence diffusant l'événement. Ce code affiche la valeur de la propriété cible dans le champ d'entrée du texte.

Syntaxe d'événement supplémentaire

Hormis un objet d'écoute, vous pouvez utiliser une fonction en tant qu'écouteur. Un écouteur est une fonction si elle n'appartient pas à un objet. Par exemple, le code suivant crée la fonction écouteur `monGestionnaire` et l'enregistre sur `occurrenceDeBouton` :

```
function monGestionnaire(objEvnt){
  if (objEvnt.type == "click"){
    // votre code ici
  }
}
occurrenceDeBouton.addEventListener("click", monGestionnaire);
```

Remarque : Dans l'écouteur d'une fonction, le mot-clé `this` est `buttonInstance`, et non le scénario sur lequel la fonction est définie.

Vous pouvez également utiliser des objets d'écoute supportant une méthode `handleEvent`. Quel que soit le nom de l'événement, la méthode `handleEvent` de l'objet d'écoute est appelée. Vous devez utiliser une instruction `if else` ou `switch` pour traiter plusieurs événements, ce qui rend cette syntaxe maladroite. Par exemple, le code suivant utilise une instruction `if else` pour traiter les événements `click` et `enter` :

```
monObjet.handleEvent = function(o){
  if (o.type == "click"){
    // votre code ici
  } else if (o.type == "enter"){
    // votre code ici
  }
}
target.addEventListener("click", monObjet);
target2.addEventListener("enter", monObjet);
```

Il existe un autre style de syntaxe d'événement qui doit être uniquement utilisé lorsque vous créez un composant et que vous savez qu'un objet particulier est le seul écouteur pour un événement. Dans ce cas, vous pouvez tirer profit du fait que le modèle d'événement v2 appelle toujours une méthode sur l'occurrence de composant qui correspond au nom d'événement plus « Gestionnaire ». Par exemple, si vous voulez traiter l'événement `click`, vous devez écrire le code suivant :

```
occurrenceDeComposant.clickHandler = function(o){
  // insérez votre code ici
}
```

Dans le code ci-dessus, le mot-clé `this`, s'il est utilisé dans la fonction de rappel, a un domaine limité à `occurrenceDeComposant`.

Pour plus d'informations, consultez *Création de composants*. Il vous faudra parfois télécharger le fichier PDF le plus récent sur le site du Centre de support Flash pour obtenir ces informations.

Création de la navigation personnalisée du focus

Lorsqu'un utilisateur appuie sur la touche de tabulation pour naviguer dans une application Flash ou qu'il clique dans une application, la *Classe FocusManager* détermine le composant qui reçoit du focus. Vous n'avez pas besoin d'ajouter une occurrence FocusManager à une application ou de rédiger du code pour activer FocusManager.

Si un objet RadioButton reçoit du focus, FocusManager examine cet objet et tous les objets ayant la même valeur `groupName`. FocusManager définit alors le focus sur l'objet dont la propriété `selected` est définie sur `true`.

Chaque composant Window modal contient une occurrence de FocusManager pour que les commandes de la fenêtre définissent eux-mêmes la tabulation afin d'empêcher l'utilisateur d'aller dans les composants des autres fenêtres avec la touche de tabulation.

Pour créer une navigation de focus dans une application, définissez la propriété `tabIndex` sur tous les composants (y compris les boutons) qui doivent recevoir du focus. Lorsqu'un utilisateur appuie sur la touche de tabulation, la *Classe FocusManager* cherche un objet activé dont une propriété `tabIndex` est supérieure à la valeur actuelle de `tabIndex`. Une fois que la *Classe FocusManager* a trouvé la propriété `tabIndex` la plus élevée, il retombe à zéro. Dans l'exemple suivant, l'objet `comment` (probablement un composant TextArea) reçoit le focus le premier, puis c'est au tour de l'objet `okButton` :

```
comment.tabIndex = 1;
okButton.tabIndex = 2;
```

Pour créer un bouton qui reçoive le focus lorsqu'un utilisateur appuie sur Entrée (Windows) ou sur Retour (Macintosh), définissez la propriété *`FocusManager.defaultPushButton`* sur le nom d'occurrence du bouton désiré, comme dans l'exemple suivant :

```
FocusManager.defaultPushButton = okButton;
```

La *Classe FocusManager* remplace le rectangle de focus par défaut de Flash Player et trace un rectangle de focus personnalisé avec des bords arrondis.

Gestion de la profondeur des composants dans un document

Si vous voulez positionner un composant au-dessus ou au-dessous d'un autre objet dans une application, vous devez utiliser la *Classe DepthManager*. L'API DepthManager vous permet de placer les composants d'interface utilisateur dans un ordre z approprié (par exemple, une liste déroulante se déroule au-dessus d'autres composants, des curseurs apparaissent devant tous les objets, des fenêtres de dialogue flottent sur le contenu, etc.).

DepthManager a deux objectifs principaux : gérer les affectations de profondeur relatives dans un document et gérer les profondeurs réservées sur le scénario racine pour les services système tels que le curseur et les info-bulles.

Pour utiliser DepthManager, appelez ses méthodes (consultez *Classe DepthManager*, page 98).

Le code suivant place l'occurrence de composant `loader` sous le composant `Button` :

```
loader.setDepthBelow(button);
```

A propos de l'utilisation d'un preloader avec les composants

Les composants sont définis sur Exporter dans la première image par défaut. Cela signifie que les composants sont chargés avant que la première image d'une application ne soit rendue. Si vous voulez créer un preloader pour une application, vous devez désélectionner Exporter dans la première image pour tous les symboles de clips compilés dans votre bibliothèque.

Remarque : Si vous utilisez le composant ProgressBar pour afficher la progression du chargement, laissez l'option Exporter dans la première image sélectionnée pour ProgressBar.

Mise à niveau des composants v1 vers l'architecture v2

Les composants v2 ont été rédigés pour être conformes à plusieurs normes du web (en matière d'événements, de styles, d'instructions de lecture/définitions, etc.) et sont très différents des composants v1 de Macromedia Flash MX et des DRK antérieurs à Macromedia Flash MX 2004. Les composants v2 ont des API différentes et ont été rédigés dans ActionScript 2. L'utilisation de composants v1 et v2 dans une même application peut donc entraîner un comportement imprévisible. Pour plus d'informations sur la mise à niveau des composants v1 afin d'utiliser le traitement d'événements, les styles de la version 2 et l'accès de lecture/définitions aux propriétés au lieu des méthodes, consultez *Création de composants*. Il vous faudra parfois télécharger le fichier PDF le plus récent sur le site du Centre de support Flash pour obtenir ces informations.

Les applications Flash contenant des composants v1 fonctionnent correctement dans Flash Player 6 et Flash Player 7, lorsqu'elles sont publiées pour Flash Player 6 ou 6r65. Si vous souhaitez mettre vos applications à jour afin de pouvoir travailler avec des publications pour Flash Player 7, vous devez convertir votre code afin d'utiliser la saisie stricte des données. Pour plus d'informations, consultez « Création des classes avec ActionScript 2 », dans le Guide de référence ActionScript de l'aide.

CHAPITRE 3
Personnalisation des composants

Si vous utilisez les mêmes composants dans des applications différentes, il est possible que vous souhaitiez modifier leur aspect. Il existe trois manières de le faire dans Macromedia Flash MX 2004 et Macromedia Flash MX Professionnel 2004 :

- utiliser l'API des styles ;
- appliquer un thème ;
- modifier ou remplacer les enveloppes d'un composant.

L'API (interface de programmation) des styles contient des méthodes et des propriétés qui permettent de modifier la couleur et la mise en forme du texte du composant.

Un thème correspond à l'ensemble des styles et des enveloppes qui constituent l'aspect d'un composant.

Les enveloppes sont des symboles utilisés pour afficher les composants. *L'application d'une enveloppe* est le processus mis en œuvre pour changer l'aspect d'un composant en modifiant ou en remplaçant ses graphiques sources. Une enveloppe peut correspondre à un petit objet, comme l'extrémité d'une bordure ou un coin, ou à un objet composite comme l'image complète d'un bouton dans son état relevé (qui indique que personne n'a appuyé dessus). Une enveloppe peut également être un symbole sans graphique, qui contient un code traçant un objet du composant.

Utilisation des styles pour personnaliser la couleur et le texte des composants

Toutes les occurrences d'un composant ont des propriétés de style et des méthodes `setStyle()` et `getStyle()` (consultez *`UIObject.setStyle()`* et *`UIObject.getStyle()`*) que vous pouvez utiliser pour accéder aux propriétés de style et les modifier. Vous pouvez utiliser des styles pour personnaliser un composant de plusieurs manières :

- Définir des styles sur une occurrence de composant.

  Vous pouvez modifier les propriétés de couleur et de texte d'une seule occurrence de composant. Cette méthode est efficace dans certaines situations, mais elle peut prendre du temps si vous devez définir des propriétés individuelles pour tous les composants d'un document.
- Utiliser la déclaration de style _global qui définit les styles de tous les composants d'un document.

  Si vous voulez appliquer le même aspect à l'ensemble du document, vous pouvez créer des styles sur la déclaration de style _global.

- Créer des déclarations de style personnalisées et les appliquer à des occurrences de composant spécifiques.

  Vous pouvez également faire en sorte que des groupes de composants partagent un même style dans un document. Pour ce faire, vous pouvez créer des déclarations de style personnalisées à appliquer à des composants spécifiques.
- Créer des déclarations de style de classe par défaut.

  Vous pouvez également définir une déclaration de style de classe par défaut de sorte que toutes les occurrences d'une classe partagent un aspect par défaut.

Les modifications apportées aux propriétés de style ne sont pas affichées lors de la visualisation des composants sur la scène au moyen de la fonction d'aperçu en direct. Pour plus d'informations, consultez *Composants dans l'aperçu en direct*, page 17.

Définition de styles pour l'occurrence d'un composant

Vous pouvez rédiger un code ActionScript pour définir et appliquer des propriétés de style à n'importe quelle occurrence d'un composant. Les méthodes `UIObject.setStyle()` et `UIObject.getStyle()` peuvent être appelées directement à partir de n'importe quel composant. Par exemple, le code suivant définit la couleur du texte d'une occurrence Button appelée `monBouton` :

```
monBouton.setStyle("color", 0xFF00FF");
```

Bien qu'il soit possible d'accéder aux styles en tant que propriétés (par exemple, `monBouton.color = 0xFF00FF`), il est préférable d'utiliser les méthodes `setStyle()` et `getStyle()` pour assurer le bon fonctionnement des styles. Pour plus d'informations, consultez *Définition des valeurs des propriétés de style*, page 33.

Remarque : Vous ne devez pas appeler la méthode *UIObject.setStyle()* plusieurs fois pour définir plusieurs propriétés. Si vous voulez modifier plusieurs propriétés, ou modifier les propriétés de plusieurs occurrences d'un composant, créez un format de style personnalisé. Pour plus d'informations, consultez *Définition des styles pour des composants spécifiques*, page 29.

Pour définir ou changer une propriété pour une occurrence de composant :

1. Sélectionnez l'occurrence de composant sur la scène.
2. Dans l'inspecteur des propriétés, donnez-lui le nom d'occurrence **monComp**.
3. Ouvrez le panneau Actions et choisissez Scène 1, puis Calque 1 : Image 1.
4. Saisissez le code suivant pour appliquer la couleur bleue à l'occurrence :

   ```
   monComp.setStyle("themeColor", "haloBlue");
   ```

   La syntaxe suivante définit une propriété et une valeur pour l'occurrence d'un composant :

   ```
   instanceName.setStyle("propriété", value);
   ```
5. Choisissez Contrôle > Tester l'animation pour visualiser les modifications.

   Pour obtenir la liste des styles supportés, consultez *Styles supportés*, page 34.

Définition des styles globaux

La déclaration de style _global est affectée à tous les composants de l'interface utilisateur Flash avec la version 2 de l'architecture des composants Macromedia (composants v2). L'objet _global a une propriété intitulée `style` (`_global.style`). Il s'agit d'une occurrence de CSSStyleDeclaration. Cette propriété `style` agit comme la déclaration de style _global. Si vous modifiez la valeur d'une propriété sur la déclaration de style _global, la modification s'applique à tous les composants de votre document Flash.

Certains styles sont définis sur l'objet CSSStyleDeclaration d'une classe de composants (par exemple, le style `backgroundColor` des composants TextArea et TextInput). La déclaration de style de classe étant prioritaire par rapport à la déclaration de style _global lors de la détermination des valeurs de style, la définition du style `backgroundColor` sur la déclaration de style _global n'a aucun effet sur TextArea et TextInput. Pour plus d'informations, consultez *Utilisation des styles globaux, personnalisés et de classe dans le même document*, page 31.

Pour modifier une ou plusieurs propriétés de la déclaration de style global :

1. Assurez-vous que le document contient au moins une occurrence de composant.

   Pour plus d'informations, consultez *Ajout de composants aux documents Flash*, page 18.
2. Créez et nommez un calque dans le scénario.
3. Choisissez une image dans laquelle le composant apparaît (ou avant qu'il n'apparaisse), dans le nouveau calque.
4. Ouvrez le panneau Actions.
5. Utilisez la syntaxe suivante pour modifier les propriétés sur la déclaration de style _global. Vous devez uniquement répertorier les propriétés dont vous voulez modifier les valeurs, comme dans l'exemple suivant :

   ```
   _global.style.setStyle("color", 0xCC6699);
   _global.style.setStyle("themeColor", "haloBlue")
   _global.style.setStyle("fontSize",16);
   _global.style.setStyle("fontFamily" , "_serif");
   ```

   Pour une liste des styles, consultez *Styles supportés*, page 34.
6. Choisissez Contrôle > Tester l'animation pour visualiser les modifications.

Définition des styles pour des composants spécifiques

Vous pouvez créer des déclarations de style personnalisées pour définir un jeu unique de propriétés pour des composants spécifiques de votre document Flash. Créez une occurrence de l'objet CSSStyleDeclaration, créez un nom de style personnalisé et placez-le sur la liste `_global.styles` (`_global.styles.newStyle`). Indiquez les propriétés et valeurs de ce style et affectez le style à une occurrence. L'objet CSSStyleDeclaration est accessible si vous avez placé au moins une occurrence de composant sur la scène.

Pour apporter des modifications à un format de style personnalisé, procédez de la même manière que pour modifier les propriétés de la déclaration de style _global. Au lieu du nom de la déclaration de style _global, utilisez l'occurrence CSSStyleDeclaration. Pour plus d'informations sur la déclaration de style _global, consultez *Définition des styles globaux*, page 29.

Pour plus d'informations sur les propriétés de l'objet CSSStyleDeclaration, consultez *Styles supportés*, page 34. Pour une liste des styles supportés par chacun des composants, consultez leurs entrées respectives dans le Chapitre 4, *Dictionnaire des composants*, page 47.

Pour créer une déclaration de style personnalisée pour des composants spécifiques :

1 Assurez-vous que le document contient au moins une occurrence de composant.

Pour plus d'informations, consultez *Ajout de composants aux documents Flash*, page 18.

Cet exemple utilise trois composants button avec les noms d'occurrence `a`, `b` et `c`. Si vous utilisez des composants différents, donnez-leur des noms d'occurrence dans l'inspecteur des propriétés et utilisez ces noms à l'étape 9.

2 Créez et nommez un calque dans le scénario.

3 Choisissez une image dans laquelle le composant apparaît (ou avant qu'il n'apparaisse), dans le nouveau calque.

4 Ouvrez le panneau Actions en mode Expert.

5 Utilisez la syntaxe suivante pour créer une occurrence de l'objet CSSStyleDeclaration et définir le nouveau format de style personnalisé :

```
var styleObj = new mx.styles.CSSStyleDeclaration;
```

6 Définissez la propriété `styleName` de la déclaration de style pour donner un nom au style :

```
styleObj.styleName = "newStyle";
```

7 Placez le style sur la liste des styles globaux :

```
_global.styles.newStyle = styleObj;
```

Remarque : Vous pouvez également créer un objet CSSStyleDeclaration et l'affecter à une nouvelle déclaration de style en utilisant la syntaxe suivante :

```
var styleObj = _global.styles.newStyle = new
  mx.styles.CSSStyleDeclaration();
```

8 Utilisez la syntaxe suivante pour spécifier les propriétés à définir pour la déclaration de style `monStyle` :

```
styleObj.fontFamily = "_sans";
styleObj.fontSize = 14;
styleObj.fontWeight = "bold";
styleObj.textDecoration = "underline";
styleObj.color = 0x336699;
styleObj.setStyle("themeColor", "haloBlue");
```

9 Dans la même fenêtre de script, utilisez la syntaxe suivante pour définir la propriété `styleName` des deux composants spécifiques sur la déclaration de style personnalisée :

```
a.setStyle("styleName", "newStyle");
b.setStyle("styleName", "newStyle");
```

Vous pouvez également accéder aux styles d'une déclaration de style personnalisée en utilisant les méthodes `setStyle()` et `getStyle()`. Le code suivant définit le style `backgroundColor` sur la déclaration de style `newStyle` :

```
_global.styles.newStyle.setStyle("backgroundColor", "0xFFCCFF");
```

Définition des styles pour une classe de composants

Vous pouvez définir une déclaration de style de classe pour n'importe quelle classe de composants (Button, CheckBox, etc.) qui définit des styles par défaut pour toutes les occurrences de cette classe. Vous devez créer la déclaration de style avant de créer les occurrences. Certains composants, tels que TextArea et TextInput, ont des déclarations de style de classe prédéfinies par défaut, car leurs propriétés `borderStyle` et `backgroundColor` doivent être personnalisées.

Le code suivant crée une déclaration de style de classe pour CheckBox et applique la couleur bleue à la case à cocher :

```
var o = _global.styles.CheckBox = new mx.styles.CSSStyleDeclaration();
o.color = 0x0000FF;
```

Vous pouvez également accéder aux styles d'une déclaration de style personnalisée en utilisant les méthodes `setStyle()` et `getStyle()`. Le code suivant définit le style de couleur sur la déclaration de style RadioButton :

```
_global.styles.RadioButton.setStyle("color", "blue");
```

Pour plus d'informations sur les styles supportés, consultez *Styles supportés*, page 34.

Utilisation des styles globaux, personnalisés et de classe dans le même document

Si vous définissez un style à un seul emplacement du document, Flash se sert de cette définition pour connaître la valeur d'une propriété. Cependant, il arrive qu'un document Flash ait une déclaration de style _global, des déclarations de style personnalisées, des propriétés de style définies directement sur les occurrences de composant et des déclarations de style de classe par défaut. Dans ce cas, Flash détermine la valeur d'une propriété en recherchant sa définition à tous les emplacements du document, en suivant un ordre spécifique.

Flash commence par chercher une propriété de style sur l'occurrence du composant. Si le style n'est pas défini directement sur l'occurrence, Flash examine la propriété `styleName` de l'occurrence pour voir si une déclaration de style lui a été affectée.

Si la propriété `styleName` n'a pas été affectée à une déclaration de style, Flash recherche la propriété sur une déclaration de style de classe par défaut. S'il n'y a pas de déclaration de style de classe et que la propriété n'hérite pas de sa valeur, la déclaration de style _global est cochée. Si la propriété n'est pas définie sur la déclaration de style _global, elle est `undefined`.

S'il n'y a pas de déclaration de style de classe et que la propriété n'hérite pas de sa valeur, Flash recherche la propriété sur le parent de l'occurrence. Si la propriété n'est pas définie sur le parent, Flash vérifie la propriété `styleName` du parent ; si elle n'est pas définie, Flash continue de parcourir les occurrences de parent jusqu'au niveau _global. Si la propriété n'est pas définie sur la déclaration de style _global, elle est `undefined`.

StyleManager informe Flash si un style hérite de sa valeur ou non. Pour plus d'informations, consultez *Classe StyleManager*, page 205.

Remarque : La valeur CSS `"inherit"` n'est pas supportée.

A propos des propriétés de style de couleur

Les propriétés de style de couleur sont différentes de celles des autres styles. Toutes les propriétés de couleur ont un nom se terminant par « Color », par exemple `backgroundColor`, `disabledColor` et `color`. Une fois les propriétés de style de couleur modifiées, la couleur change instantanément dans l'occurrence et dans toutes les occurrences enfants appropriées. Toutes les autres modifications des propriétés de style ne font que marquer l'objet pour indiquer qu'il doit être retracé et que les modifications ne seront appliquées qu'à l'image suivante.

La valeur d'une propriété de style de couleur peut être un chiffre, une chaîne ou un objet. S'il s'agit d'un chiffre, il représente la valeur RVB de la couleur en tant que nombre hexadécimal (0xRRGGBB). Si la valeur est une chaîne, il doit s'agir d'un nom de couleur.

Ces noms sont des chaînes qui correspondent aux couleurs couramment utilisées. De nouveaux noms de couleur peuvent être ajoutés à l'aide de StyleManager (consultez *Classe StyleManager*, page 205). Le tableau suivant répertorie les noms de couleur par défaut :

Nom de couleur	Valeur
noir	0x000000
blanc	0xFFFFFF
rouge	0xFF0000
vert	0x00FF00
bleu	0x0000FF
magenta	0xFF00FF
jaune	0xFFFF00
cyan	0x00FFFF

Remarque : Si le nom de couleur n'est pas défini, le composant risque de ne pas être correctement tracé.

Vous pouvez utiliser n'importe quel identificateur ActionScript légal pour créer vos propres noms de couleur (par exemple « `WindowText` » ou « `ButtonText` »). Utilisez StyleManager pour définir de nouvelles couleurs, comme dans l'exemple suivant :

```
mx.styles.StyleManager.registerColorName("special_blue", 0x0066ff);
```

La plupart des composants ne peuvent pas traiter un objet en tant que valeur de propriété de style de couleur. Cependant, certains d'entre eux peuvent traiter des objets de couleur qui représentent des dégradés ou autres combinaisons. Pour plus d'informations, consultez la section « Utilisation des styles » des entrées de composant respectives dans le Chapitre 4, *Dictionnaire des composants*, page 47.

Vous pouvez utiliser des déclarations de style de classe et des noms de couleur pour contrôler facilement les couleurs du texte et des symboles à l'écran. Par exemple, si vous voulez un écran de configuration de l'affichage qui ressemble à Microsoft Windows, vous devez définir des noms de couleur tels que ButtonText et WindowText et des déclarations de style de classe telles que Button, CheckBox et Window. En définissant sur ButtonText et WindowText les propriétés de style de couleur sur les déclarations de style et en fournissant une interface utilisateur permettant de modifier les valeurs de ButtonText et WindowText, vous pouvez fournir les mêmes modèles de couleurs que Microsoft Windows, le système d'exploitation Macintosh ou tout autre système d'exploitation.

Définition des valeurs des propriétés de style

Utilisez la méthode *`UIObject.setStyle()`* pour définir une propriété de style sur l'occurrence d'un composant, la déclaration de style global, une déclaration de style personnalisée ou une déclaration de style de classe. Le code suivant définit la couleur rouge pour le style `color` d'une occurrence de bouton radio :

```
monBoutonRadio.setStyle("color", "red");
```

Le code suivant définit le style `color` de la déclaration de style personnalisée CheckBox :

```
_global.styles.CheckBox.setStyle("color", "white");
```

La méthode *`UIObject.setStyle()`* détermine si un style est hérité et notifie les enfants de cette occurrence si leur style change. Elle notifie également l'occurrence à retracer pour refléter le nouveau style. Vous devez donc utiliser `setStyle()` pour définir ou modifier les styles. Cependant, pour optimiser la création de déclarations de style, vous pouvez définir directement les propriétés sur un objet. Pour plus d'informations, consultez *Définition des styles globaux*, page 29, *Définition des styles pour une classe de composants*, page 31 et *Définition des styles pour des composants spécifiques*, page 29.

Utilisez la méthode *`UIObject.getStyle()`* pour définir une propriété de style sur l'occurrence d'un composant, la déclaration de style global, une déclaration de style personnalisée ou une déclaration de style de classe. Le code suivant obtient la valeur de la propriété de couleur et l'affecte à la variable `o` :

```
var o = monBoutonRadio.getStyle("color");
```

Le code suivant obtient la valeur d'une propriété de style définie sur la déclaration de style _global :

```
var r = _global.style.getValue("marginRight");
```

Si le style n'est pas défini, `getStyle()` peut renvoyer la valeur `undefined`. Cependant, `getStyle()` comprend le fonctionnement de l'héritage des propriétés de style. Cela signifie que même si les styles sont des propriétés, vous devez utiliser *`UIObject.getStyle()`* pour y accéder et ne pas avoir besoin de savoir si le style est hérité ou non.

Pour plus d'informations, consultez *`UIObject.getStyle()`*, et *`UIObject.setStyle()`*.

Styles supportés

Flash MX 2004 et Flash MX Professionnel 2004 ont deux thèmes : *Halo* (HaloTheme.fla) et *Echantillon* (SampleTheme.fla). Ces thèmes supportent un ensemble de styles différent. Le thème Echantillon utilise tous les styles du mécanisme de styles v2. Il vous permet de visualiser un échantillon des styles contenus dans un document. Le thème Halo supporte un sous-ensemble des styles du thème Echantillon.

Les propriétés de style suivantes sont supportées par la plupart des composants v2 dans le style Echantillon. Pour plus d'informations sur les styles Halo supportés par les différents composants, consultez le Chapitre 4, *Dictionnaire des composants*, page 47.

Si des valeurs non autorisées sont saisies, la valeur par défaut est utilisée. Ce facteur est important si vous réutilisez des déclarations de style CSS qui utilisent des valeurs externes au sous-ensemble de valeurs Macromedia.

Les composants peuvent supporter les styles suivants :

Style	Description
backgroundColor	Arrière-plan d'un composant. Il s'agit du seul style de couleur qui n'hérite pas de sa valeur. La valeur par défaut est la transparence.
borderColor	Section noire de la bordure à trois dimensions ou la section de couleur d'une bordure à deux dimensions. La valeur par défaut est 0x000000 (noir).
borderStyle	Bordure du composant : « aucune », « incrusté », « relief » ou « uni ». Ce style n'hérite pas de sa valeur. La valeur par défaut est « uni ».
buttonColor	Surface d'un bouton et section d'une bordure à trois dimensions. La valeur par défaut est 0xEFEEEF (gris clair).
color	Texte inscrit sur l'étiquette d'un composant. La valeur par défaut est 0x000000 (noir).
disabledColor	Couleur désactivée pour le texte. La couleur par défaut est 0x848384 (gris foncé).
fontFamily	Nom de police pour le texte. La valeur par défaut est _sans.
fontSize	Taille du point pour la police. La valeur par défaut est 10.
fontStyle	Style de police : « normal » ou « italique ». La valeur par défaut est « normal ».
fontWeight	Epaisseur de la police : « normal » ou « gras ». La valeur par défaut est « normal ».
highlightColor	Section de la bordure à trois dimensions. La valeur par défaut est 0xFFFFFF (blanc).
marginLeft	Nombre indiquant la marge gauche pour le texte. La valeur par défaut est 0.
marginRight	Nombre indiquant la marge droite pour le texte. La valeur par défaut est 0.

Style	Description
scrollTrackColor	Piste de défilement pour une barre de défilement. La valeur par défaut est 0xEFEEEF (gris clair).
shadowColor	Section de la bordure à trois dimensions. La valeur par défaut est 0x848384 (gris foncé).
symbolBackgroundColor	Couleur d'arrière-plan des cases à cocher et des boutons radio. La valeur par défaut est 0xFFFFFF (blanc).
symbolBackgroundDisabledColor	Couleur d'arrière-plan des cases à cocher et des boutons radio désactivés. La valeur par défaut est 0xEFEEEF (gris clair).
symbolBackgroundPressedColor	Couleur d'arrière-plan des cases à cocher et des boutons radio enfoncés. La valeur par défaut est 0xFFFFFF (blanc).
symbolColor	Coche de la case à cocher ou point du bouton radio. La valeur par défaut est 0x000000 (noir).
symbolDisabledColor	Couleur de la coche ou du point de bouton radio désactivés. La valeur par défaut est 0x848384 (gris foncé).
textAlign	Alignement du texte : « gauche », « droite » ou « centré ». La valeur par défaut est « gauche ».
textDecoration	Décoration du texte : « aucune » ou « souligné ». La valeur par défaut est « aucun ».
textIndent	Nombre indiquant le retrait du texte. La valeur par défaut est 0.

A propos des thèmes

Les thèmes sont des collections de styles et d'enveloppes. Le thème par défaut pour Flash MX 2004 et Flash MX Professionnel 2004 est Halo (HaloTheme.fla). Le thème Halo a été développé pour garantir à vos utilisateurs une utilisation précise et fiable de vos applications. Flash MX 2004 et Flash MX Professionnel 2004 inclut un thème supplémentaire intitulé Echantillon (SampleTheme.fla). Le thème Echantillon vous permet d'essayer le jeu complet des styles disponibles pour les composants v2 (le thème Halo utilise uniquement un sous-ensemble des styles disponibles). Les fichiers de thème sont situés dans les dossiers suivants :

- Windows—First Run\ComponentFLA
- Macintosh—First Run/ComponentFLA

Vous pouvez créer de nouveaux thèmes et les utiliser dans une application pour modifier l'aspect de tous les composants. Vous pouvez par exemple créer un thème à deux dimensions et un thème à trois dimensions.

Les composants v2 utilisent des enveloppes (symboles graphiques ou de clip) pour afficher leurs apparences visuelles. Le fichier .as associé à chaque composant contient un code qui charge des enveloppes spécifiques pour le composant. Par exemple, ScrollBar est programmé pour charger le symbole accompagné de l'identificateur de liaison `DownArrowDown` comme enveloppe pour l'état enfoncé (bas) de sa flèche bas. Vous pouvez facilement créer un thème en effectuant une copie du thème Halo ou Echantillon et en modifiant les graphiques dans les enveloppes.

Un thème peut également contenir un nouveau jeu de styles. Vous devez rédiger le code ActionScript pour créer une déclaration de style global et des déclarations de style supplémentaires. Pour plus d'informations, consultez *Utilisation des styles pour personnaliser la couleur et le texte des composants*, page 27.

Application d'un thème à un document

Pour appliquer un nouveau thème à un document, ouvrez un fichier FLA de thème en tant que bibliothèque externe et faites glisser le dossier Thème de la bibliothèque externe sur la scène. Les étapes suivantes expliquent le processus en détail.

Pour appliquer un thème à un document :

1. Choisissez Fichier > Ouvrir et ouvrez le document qui utilise des composants v2 dans Flash ou choisissez Fichier > Nouveau et créez un document qui utilise des composants v2.
2. Sélectionnez Fichier > Enregistrer et choisissez un nom unique, tel que **ThemeApply.fla**.
3. Choisissez Fichier > Importer > Ouvrir une bibliothèque externe et sélectionnez le fichier FLA du thème à appliquer à votre document.

   Si vous n'avez pas créé de thème, vous pouvez utiliser le thème Echantillon, situé dans le dossier Flash 2004/en/Configuration/SampleFLA.
4. Dans le panneau Bibliothèque du thème, choisissez Flash UI Components 2 > Themes > MMDefault et faites glisser le dossier Assets de tous les composants de votre document vers la bibliothèque ThemeApply.fla.

   Si vous ne savez pas exactement quels composants sont contenus dans les documents, vous pouvez faire glisser l'ensemble du dossier Thèmes vers la scène. Les enveloppes situées dans le dossier Thèmes de la bibliothèque sont automatiquement affectées aux composants du document.

   Remarque : L'aperçu en direct des composants sur la scène ne reflète pas le nouveau thème.
5. Choisissez Contrôle > Tester l'animation pour visualiser le document sur lequel est appliqué le nouveau thème.

Création d'un thème

Si vous ne voulez pas utiliser le thème Halo ou le thème Echantillon, vous pouvez les modifier pour créer un nouveau thème.

Certaines enveloppes ont une taille prédéfinie dans les thèmes. Vous pouvez en augmenter ou en réduire la taille. Les composants sont alors automatiquement redimensionnés à leur nouvelle taille. Les autres enveloppes sont composées de plusieurs éléments, certains statiques et d'autres extensibles.

Certaines enveloppes (par exemple, RectBorder et ButtonSkin) utilisent l'API ActionScript Drawing pour tracer leurs graphiques, car elle est beaucoup plus efficace en termes de taille et de performances. Vous pouvez vous servir du code ActionScript comme modèle dans ces enveloppes afin de les adapter à vos besoins.

Pour créer un thème :

1 Sélectionnez le fichier FLA correspondant au thème à utiliser comme modèle et faites-en une copie.

Donnez un nom unique à la copie, tel que « **MonThème.fla** ».

2 Choisissez Fichier > Ouvrir MonTheme.fla dans Flash.

3 Choisissez Fenêtre > Bibliothèque pour ouvrir la bibliothèque si ce n'est pas déjà fait.

4 Double-cliquez sur le symbole d'une enveloppe que vous souhaitez modifier pour l'ouvrir en mode de modification de symbole.

Les enveloppes sont situées dans le dossier Themes > MMDefault > *Composant* Assets (dans cet exemple, Themes > MMDefault > RadioButton Assets).

5 Modifiez le symbole ou supprimez les graphiques et créez-en de nouveaux.

Vous pouvez choisir Affichage > Zoom avant pour augmenter le zoom. Lors de la modification d'une enveloppe, vous devez conserver le point d'alignement pour qu'elle s'affiche correctement. Le coin supérieur gauche de tous les symboles modifiés doit se trouver à (0,0).

6 Une fois que vous avez terminé de modifier le symbole de l'enveloppe, cliquez sur le bouton de retour, du côté gauche de la barre d'information en haut de la scène, pour revenir en mode d'édition de document.

7 Répétez les étapes 4 à 6 jusqu'à ce que vous ayez modifié toutes les enveloppes voulues.

8 Appliquez MonTheme.fla à un document en suivant les étapes de la section précédente, *Application d'un thème à un document*, page 36.

A propos de l'application des enveloppes aux composants

Les enveloppes sont des symboles utilisés par les composants pour afficher leur aspect. Il s'agit de symboles graphiques ou de clip. La plupart des enveloppes contiennent des formes représentant l'aspect du composant. Certaines enveloppes contiennent uniquement le code ActionScript qui trace le composant dans le document.

Les composants v2 de Macromedia sont des clips compilés, ce qui signifie que vous ne pourrez pas visualiser leurs actifs dans la bibliothèque. Cependant, les fichiers FLA sont installés avec Flash et contiennent toutes les enveloppes des composants. Ces fichiers FLA sont appelés des *thèmes*. Tous les thèmes ont des aspects et des comportements différents, mais ils contiennent tous des enveloppes ayant les mêmes noms de symbole et les mêmes identificateurs de liaison. Cela permet de faire glisser un thème vers la scène du document pour en modifier l'aspect. Pour plus d'informations sur les thèmes, consultez *A propos des thèmes*, page 35. Les fichiers FLA de thème sont également utilisés pour modifier les enveloppes des composants. Les enveloppes sont situées dans le dossier Thèmes, dans le panneau Bibliothèque de tous les fichiers FLA de thème.

Chaque composant comporte de nombreuses enveloppes. Par exemple, la flèche bas du composant ScrollBar se compose de trois enveloppes : ScrollDownArrowDisabled, ScrollDownArrowUp et ScrollDownArrowDown. Certains composants partagent des enveloppes. Les composants qui utilisent des barres de défilement, comme ComboBox, List, ScrollBar et ScrollPane, partagent les enveloppes du dossier ScrollBar Skins. Vous pouvez modifier les enveloppes existantes et en créer de nouvelles pour changer l'aspect des composants.

Le fichier .as qui définit chaque classe de composant contient un code qui charge des enveloppes spécifiques destinées au composant. Chaque enveloppe de composant a une propriété d'enveloppe affectée à l'identificateur de liaison du symbole d'une enveloppe. Par exemple, l'état enfoncé (bas) de la flèche bas de ScrollBar porte le nom de propriété d'enveloppe `downArrowDownName`. La valeur par défaut de la propriété `downArrowDownName` est `« DownArrowDown »`, ce qui correspond à l'identificateur de liaison du symbole de l'enveloppe. Vous pouvez modifier les enveloppes et les appliquer à un composant en utilisant ces propriétés d'enveloppe. Il n'est pas nécessaire de modifier le fichier .as du composant pour changer les propriétés de son enveloppe ; vous pouvez passer les valeurs des propriétés de l'enveloppe à la fonction de constructeur du composant lors de la création d'un composant dans votre document.

Choisissez l'une des manières suivantes d'envelopper un composant en fonction de ce que vous souhaitez faire :

- Pour remplacer toutes les enveloppes dans un document par un nouveau jeu (avec tous les types de composant partageant le même aspect), appliquez un thème (consultez *A propos des thèmes*, page 35).

  Remarque : Cette méthode d'enveloppe est conseillée aux débutants car elle ne nécessite pas la rédaction d'un script.

- Pour utiliser des enveloppes différentes pour les multiples occurrences d'un même composant, modifiez les enveloppes existantes et définissez leurs propriétés (consultez *Modification des enveloppes des composants*, page 39 et *Application d'une enveloppe modifiée à un composant*, page 40).
- Pour modifier les enveloppes dans un sous-composant (tel qu'une barre de défilement ou dans un composant List), sous-classez le composant (consultez *Application d'une enveloppe modifiée à un sous-composant*, page 41).
- Pour modifier les enveloppes d'un sous-composant qui ne sont pas directement accessibles à partir du composant principal (par ex. un composant List dans un composant ComboBox), remplacez les propriétés des enveloppes dans le prototype (consultez *Modification des propriétés d'enveloppe dans le prototype*, page 44).

Remarque : Les méthodes ci-dessus sont répertoriées dans l'ordre de leur facilité d'utilisation.

Modification des enveloppes des composants

Si vous voulez utiliser une enveloppe particulière pour l'occurrence d'un composant et une autre enveloppe pour une autre occurrence du composant, vous devez ouvrir un fichier FLA de thème et créer un nouveau symbole d'enveloppe. Les composants sont conçus de manière à faciliter l'utilisation des différentes enveloppes pour les différentes occurrences.

Pour modifier une enveloppe, procédez comme suit :

1 Choisissez Fichier > Ouvrir et ouvrez le fichier FLA de thème à utiliser comme modèle.

2 Choisissez Fichier > Enregistrer sous et sélectionnez un nom unique tel que **MonTheme.fla**.

3 Choisissez la ou les enveloppes à modifier (dans cet exemple, RadioTrueUp).

Les enveloppes sont situées dans le dossier Themes > MMDefault > *Composant* Assets (dans cet exemple, Themes > MMDefault > RadioButton Assets > States).

4 Choisissez Dupliquer dans le menu des options de la bibliothèque (ou en cliquant avec le bouton droit sur le symbole) et donnez un nom unique au symbole, tel que monBoutonRadioRelevé.

5 Choisissez le bouton Avancé dans la boîte de dialogue Propriétés du symbole et cochez Exporter pour ActionScript.

Un identificateur de liaison correspondant au nom du symbole est saisi automatiquement.

6 Double-cliquez sur la nouvelle enveloppe dans la bibliothèque pour l'ouvrir en mode de modification de symbole.

7 Modifiez le clip ou supprimez-le avant d'en créer un nouveau.

Vous pouvez choisir Affichage > Zoom avant pour augmenter le zoom. Lors de la modification d'une enveloppe, vous devez conserver le point d'alignement pour qu'elle s'affiche correctement. Le coin supérieur gauche de tous les symboles modifiés doit se trouver à (0,0).

8 Une fois que vous avez terminé de modifier le symbole de l'enveloppe, cliquez sur le bouton de retour, du côté gauche de la barre d'information en haut de la scène, pour revenir en mode d'édition de document.

9 Choisissez Fichier > Enregistrer, mais ne fermez pas le fichier MonTheme.fla. Vous devez maintenant créer un document dans lequel l'enveloppe modifiée sera appliquée à un composant.

Pour plus d'informations, consultez *Application d'une enveloppe modifiée à un composant*, page 40, *Modification des propriétés d'enveloppe dans le prototype*, page 44 et *Application d'une enveloppe modifiée à un sous-composant*, page 41. Pour plus d'informations sur la manière d'appliquer une nouvelle enveloppe, consultez *A propos de l'application des enveloppes aux composants*, page 37.

Remarque : Les modifications apportées aux enveloppes des composants ne sont pas affichées lors de la visualisation en aperçu direct des composants sur la scène.

Application d'une enveloppe modifiée à un composant

Une fois que vous avez modifié une enveloppe, vous devez l'appliquer au composant d'un document. Vous pouvez utiliser la méthode `createClassObject()` pour créer dynamiquement les occurrences d'un composant ou les placer manuellement sur la scène. Il existe deux moyens d'appliquer des enveloppes aux occurrences de composant, en fonction de la manière dont vous ajoutez les composants à un document.

Pour créer dynamiquement un composant et lui appliquer une enveloppe modifiée, procédez comme suit :

1 Choisissez Fichier > Nouveau pour créer un document Flash.

2 Choisissez Fichier > Enregistrer et donnez-lui un nom unique tel que **HabillageDynamique.fla**.

3 Faites glisser les composants du panneau Composants vers la scène, y compris le composant dont vous avez modifié l'enveloppe (dans cet exemple, RadioButton) et supprimez-les.

Cette action permet d'ajouter les symboles dans la bibliothèque, mais elle ne permet pas de les rendre visibles dans le document.

4 Faites glisser monBoutonRadioRelevé et tous les autres symboles personnalisés du fichier MonTheme.fla vers la scène de HabillageDynamique.fla et supprimez-les.

Cette action permet d'ajouter les symboles dans la bibliothèque, mais elle ne permet pas de les rendre visibles dans le document.

5 Ouvrez le panneau Actions et saisissez ce qui suit sur l'image 1 :

```
import mx.controls.RadioButton
createClassObject(RadioButton, "monBoutonRadio", 0,
  {trueUpIcon:"monBoutonRadioRelevé", label: "Mon Bouton Radio"});
```

6 Choisissez Contrôle > Tester l'animation.

Pour ajouter manuellement un composant sur la scène et lui appliquer une enveloppe modifiée, procédez comme suit :

1 Choisissez Fichier > Nouveau pour créer un document Flash.

2 Choisissez Fichier > Enregistrer et donnez-lui un nom unique tel que **HabillageManuel.fla**.

3 Faites glisser les composants du panneau Composants vers la scène, y compris le composant dont vous avez modifié l'enveloppe (dans cet exemple, RadioButton).

4 Faites glisser monBoutonRadioRelevé et tous les autres symboles personnalisés du fichier MonTheme.fla vers la scène de HabillageManuel.fla et supprimez-les.

Cette action permet d'ajouter les symboles dans la bibliothèque, mais elle ne permet pas de les rendre visibles dans le document.

5 Sélectionnez le composant RadioButton sur la scène et ouvrez le panneau Actions.

6 Joignez le code suivant à l'occurrence RadioButton :

```
onClipEvent(initialize){
  trueUpIcon = "monBoutonRadioRelevé";
}
```

7 Choisissez Contrôle > Tester l'animation.

Application d'une enveloppe modifiée à un sous-composant

Dans certaines situations, il est possible que vous souhaitiez modifier les enveloppes d'un sous-composant dans un composant, mais que les propriétés des enveloppes ne soient pas directement disponibles (par exemple s'il n'y a pas de moyen direct de modifier les enveloppes de la barre de défilement dans un composant List). Le code suivant vous permet d'accéder aux enveloppes des barres de défilement. Toutes les barres de défilement créées après l'exécution de ce code auront aussi de nouvelles enveloppes.

Si un composant est constitué de sous-composants, ceux-ci sont identifiés dans l'entrée du composant, dans le Chapitre 4, *Dictionnaire des composants*, page 47.

Pour appliquer une nouvelle enveloppe à un sous-composant, procédez comme suit :

1 Suivez les étapes de *Modification des enveloppes des composants*, page 39 mais en modifiant cette fois l'enveloppe d'une barre de défilement. Pour cet exemple, modifiez l'enveloppe ScrollDownArrowDown et donnez-lui le nouveau nom **maFlècheDéfilBasEnfoncée**.

2 Choisissez Fichier > Nouveau pour créer un document Flash.

3 Choisissez Fichier > Enregistrer et donnez-lui un nom unique tel que **ProjetSousComposant.fla**.

4 Double-cliquez sur le composant List dans le panneau Composants pour l'ajouter sur la scène et appuyez sur la touche Retour arrière pour le supprimer de la scène.

Cette action permet d'ajouter le composant dans le panneau Bibliothèque, mais elle ne permet pas de le rendre visible dans le document.

5 Faites glisser maFlècheDéfilBasEnfoncée et tous les autres symboles personnalisés du fichier MonTheme.fla vers la scène de ProjetSousComposant.fla et supprimez-les.

Cette action permet d'ajouter le composant dans le panneau Bibliothèque, mais elle ne permet pas de le rendre visible dans le document.

6 Procédez de l'une des manières suivantes :

- Si vous voulez modifier toutes les barres de défilement dans un document, saisissez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
import mx.controls.List
import mx.controls.scrollClasses.ScrollBar
ScrollBar.prototype.downArrowDownName = "maFlècheDéfilBasEnfoncée";
```

  Vous pouvez saisir le code suivant sur l'image 1 pour créer une liste de manière dynamique :

```
createClassObject(List, "maZoneDeListe", 0, {dataProvider:
  ["AL","AR","AZ", "CA","HI","ID", "KA","LA","MA"]});
```

  Vous pouvez également faire glisser un composant List de la bibliothèque vers la scène.

- Si vous voulez modifier une barre de défilement spécifique dans un document, saisissez le code suivant dans le panneau Actions, sur l'image 1 du scénario :

```
import mx.controls.List
import mx.controls.scrollClasses.ScrollBar
var oldName = ScrollBar.prototype.downArrowDownName;
ScrollBar.prototype.downArrowDownName = "maFlècheDéfilBasEnfoncée";
createClassObject(List, "maListe1", 0, {dataProvider: ["AL","AR","AZ",
  "CA","HI","ID", "KA","LA","MA"]});
maListe1.redraw(true);
ScrollBar.prototype.downArrowDownName = oldName;
```

  Remarque : Vous devez définir suffisamment de données pour que les barres de défilement apparaissent, ou bien définir la propriété `vScrollPolicy` sur `true`.

7 Choisissez Contrôle > Tester l'animation.

Vous pouvez également définir les enveloppes des sous-composants pour tous les composants d'un document en définissant la propriété de l'enveloppe sur l'objet `prototype` du composant, dans la section `#initclip` d'un symbole d'enveloppe. Pour plus d'informations sur l'objet prototype, consultez `Function.prototype` dans le système d'aide du Dictionnaire ActionScript.

Pour utiliser #initclip afin d'appliquer une enveloppe modifiée à tous les composants d'un document, procédez comme suit :

1 Suivez les étapes de *Modification des enveloppes des composants*, page 39 mais en modifiant cette fois l'enveloppe d'une barre de défilement. Pour cet exemple, modifiez l'enveloppe ScrollDownArrowDown et donnez-lui le nouveau nom **maFlècheDéfilBasEnfoncée**.

2 Choisissez Fichier > Nouveau pour créer un document Flash. Enregistrez-le sous un nom unique tel que **SkinsInitExample.fla**.

3 Choisissez le symbole maFlècheDéfilBasEnfoncée dans la bibliothèque de l'exemple de bibliothèque de thème modifié, faites-le glisser vers la scène de SkinsInitExample.fla et supprimez-le.

Cette action permet d'ajouter le symbole à la bibliothèque, mais pas de le rendre visible sur la scène.

4 Choisissez maFlècheDéfilBasEnfoncée dans la bibliothèque SkinsInitExample.fla et sélectionnez Liaison dans le menu d'options.

5 Cochez la case Exporter pour ActionScript. Cliquez sur OK.

L'option Exporter dans la première image est automatiquement sélectionnée.

6 Double-cliquez sur maFlècheDéfilBasEnfoncée dans la bibliothèque pour l'ouvrir en mode de modification de symbole.

7 Saisissez le code suivant sur l'image 1 du symbole maFlècheDéfilBasEnfoncée :

```
#initclip 10
  import mx.controls.scrollClasses.ScrollBar;
  ScrollBar.prototype.downArrowDownName = "maFlècheDéfilBasEnfoncée";
#endinitclip
```

8 Effectuez l'une des opérations suivantes pour ajouter un composant List au document :

- Faites glisser un composant List du panneau Composants jusqu'à la scène. Saisissez suffisamment de paramètres Label pour que la barre de défilement verticale apparaisse.
- Faites glisser un composant List du panneau Composants vers la scène et supprimez-le. Saisissez le code suivant sur l'image 1 du scénario principal de SkinsInitExample.fla :

```
createClassObject(mx.controls.List, "maListe1", 0, {dataProvider:
  ["AL","AR","AZ", "CA","HI","ID", "KA","LA","MA"]});
```

Remarque : Ajoutez suffisamment de données pour que la barre de défilement verticale apparaisse, ou bien définissez `vScrollPolicy` sur `true`.

L'exemple suivant explique comment envelopper un élément déjà sur la scène. Cet exemple enveloppe uniquement les composants List ; les barres de défilement TextArea ou ScrollPane ne seraient pas enveloppés.

Pour utiliser #initclip afin d'appliquer une enveloppe modifiée à des composants spécifiques dans un document, procédez comme suit :

1 Suivez les étapes de *Modification des enveloppes des composants*, page 39 mais en modifiant cette fois l'enveloppe d'une barre de défilement. Pour cet exemple, modifiez l'enveloppe ScrollDownArrowDown et donnez-lui le nouveau nom **maFlècheDéfilBasEnfoncée**.

2 Choisissez Fichier > Nouveau pour créer un document Flash.

3 Choisissez Fichier > Enregistrer et donnez-lui un nom unique tel que **TestMaBarreDéfilV.fla**.

4 Faites glisser maFlècheDéfilBasEnfoncée de la bibliothèque de thème vers la bibliothèque TestMaBarreDéfilV.fla.

5 Choisissez Insérer > Nouveau symbole et donnez-lui un nom unique tel que **maBarreDéfilV**.

6 Cochez la case Exporter pour ActionScript. Cliquez sur OK.

L'option Exporter dans la première image est automatiquement sélectionnée.

7 Saisissez le code suivant sur l'image 1 du symbole maBarreDéfilV :

```
#initclip 10
  importer maBarreDéfilV
  Object.registerClass("VScrollBar", maBarreDéfilV);
#endinitclip
```

8 Faites glisser un composant List du panneau Composants jusqu'à la scène.

9 Dans l'inspecteur des propriétés, saisissez le nombre de paramètres Label nécessaires pour faire apparaître la barre de défilement verticale.

10 Choisissez Fichier > Enregistrer.

11 Choisissez Fichier > Nouveau et créez un fichier ActionScript.

12 Saisissez le code suivant :

```
import mx.controls.VScrollBar
import mx.controls.List
class maBarreDéfilV extends VScrollBar{
  function init():Void{
    if (_parent instanceof List){
      downArrowDownName = "maFlècheDéfilBasEnfoncée";
    }
    super.init();
  }
}
```

13 Choisissez Fichier > Enregistrer et enregistrez le fichier sous **maBarreDéfilV.as**.

14 Cliquez sur un espace vide de la scène et, dans l'inspecteur des propriétés, choisissez le bouton Paramètres de publication.

15 Sélectionnez le bouton Paramètres de la version ActionScript.

16 Cliquez sur le bouton Plus pour ajouter un nouveau chemin de classe et sélectionnez le bouton Cible pour rechercher l'emplacement du fichier MaListeDéroulante.as sur votre disque dur.

17 Choisissez Contrôle > Tester l'animation.

Modification des propriétés d'enveloppe dans le prototype

Si un composant ne supporte pas directement les variables d'enveloppe, vous pouvez le sous-classer et remplacer ses enveloppes. Par exemple, le composant ComboBox ne supporte pas directement l'application d'enveloppes à son menu déroulant car ComboBox utilise un composant List comme menu déroulant.

Si un composant est constitué de sous-composants, ceux-ci sont identifiés dans l'entrée du composant, dans le Chapitre 4, *Dictionnaire des composants*, page 47.

Pour appliquer une enveloppe à un sous-composant, procédez comme suit :

1 Suivez les étapes de *Modification des enveloppes des composants*, page 39 mais en modifiant cette fois l'enveloppe d'une barre de défilement. Pour cet exemple, modifiez l'enveloppe ScrollDownArrowDown et donnez-lui le nouveau nom **maFlècheDéfilBasEnfoncée**.

2 Choisissez Fichier > Nouveau pour créer un document Flash.

3 Choisissez Fichier > Enregistrer et donnez-lui un nom unique tel que **monTestListe.fla**.

4 Faites glisser maFlècheDéfilBasEnfoncée de la bibliothèque de thème sur la scène de monTestListe.fla et supprimez-le.

Cette action permet d'ajouter le symbole à la bibliothèque, mais pas de le rendre visible sur la scène.

5 Choisissez Insertion > Nouveau symbole et donnez-lui un nom unique, tel que **MaListeDéroulante**.

6 Sélectionnez la case Exporter pour ActionScript et cliquez sur OK.

L'option Exporter dans la première image est automatiquement sélectionnée.

7 Saisissez le code suivant dans le panneau Actions, sur les actions de l'image 1 de MaListe :

```
#initclip 10
  import MaListe
  Object.registerClass("ComboBox", MaListe);
#endinitclip
```

8 Faites glisser un composant ComboBox vers la scène.

9 Dans l'inspecteur des propriétés, saisissez le nombre de paramètres Label nécessaires pour faire apparaître la barre de défilement verticale.

10 Choisissez Fichier > Enregistrer.

11 Choisissez Fichier > Nouveau et créez un fichier ActionScript (Flash Professionnel uniquement).

12 Saisissez le code suivant :

```
import mx.controls.ComboBox
import mx.controls.scrollClasses.ScrollBar
class MaListe extends ComboBox{
  function getDropdown():Object{
    var oldName = ScrollBar.prototype.downArrowDownName;
    ScrollBar.prototype.downArrowDownName = "maFlècheDéfilBasEnfoncée";
    var r = super.getDropdown();
    ScrollBar.prototype.downArrowDownName = oldName;
    return r;
  }
}
```

13 Choisissez Fichier > Enregistrer et enregistrez le fichier sous **MaListe.as**.

14 Cliquez sur un espace vide de la scène et, dans l'inspecteur des propriétés, choisissez le bouton Paramètres de publication.

15 Sélectionnez le bouton Paramètres de la version ActionScript.

16 Cliquez sur le bouton Plus pour ajouter un nouveau chemin de classe et sélectionnez le bouton Cible pour rechercher l'emplacement du fichier MaListeDéroulante.as sur votre disque dur.

17 Choisissez Contrôle > Tester l'animation.

CHAPITRE 4
Dictionnaire des composants

Ce chapitre de référence décrit tous les composants, ainsi que l'interface de programmation (API) de chacun d'entre eux.

Chaque description contient les informations suivantes :

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

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

Composants de l'interface utilisateur (IU)

Composant	Description
Composant Accordion	Jeu d'affichages verticaux se chevauchant, dont les boutons supérieurs permettent aux utilisateurs de passer d'un affichage à l'autre.
Composant Alert	Fenêtre contenant une question et des boutons pour la saisie de la réponse par l'utilisateur.
Composant Button	Bouton pouvant être redimensionné et personnalisé à l'aide d'une icône.
Composant CheckBox	Permet aux utilisateurs de faire un choix booléen (vrai ou faux).
Composant ComboBox	Permet aux utilisateurs de choisir une option dans une liste déroulante. Ce composant peut contenir un champ de texte modifiable dans le haut de la liste, permettant aux utilisateurs d'effectuer une recherche dans la liste.
Composant DateChooser	Permet aux utilisateurs de choisir une ou plusieurs dates dans un calendrier.
Composant DateField	Champ de texte modifiable à l'aide d'une icône de calendrier. Lorsqu'un utilisateur clique n'importe où dans le cadre de délimitation du composant, un composant DateChooser apparaît.

Composant	Description
Composant DataGrid	Permet aux utilisateurs d'afficher et de manipuler plusieurs colonnes de données.
Composant Label	Champ de texte d'une ligne non-modifiable.
Composant List	Permet aux utilisateurs de choisir une ou plusieurs options dans une liste déroulante.
Composant Loader	Conteneur contenant un fichier SWF ou JPEG chargé.
Composant Menu	Permet aux utilisateurs de choisir une commande dans une liste ; menu d'application de bureau standard.
Composant MenuBar	Barre de menus horizontale. Les menus agissent en tant que groupe et vous pouvez mieux gérer les saisies clavier et souris.
Composant NumericStepper	Flèches sur lesquelles vous devez cliquer pour augmenter ou réduire la valeur d'un nombre.
Composant ProgressBar	Affiche la progression d'un processus, généralement le chargement.
Composant RadioButton	Permet aux utilisateurs de choisir entre des options qui s'excluent les unes les autres.
Composant ScrollPane	Affiche des animations, des bitmaps et des fichiers SWF dans une zone délimitée à l'aide de barres de défilement automatiques.
Composant TextArea	Champ de texte à plusieurs lignes modifiable.
Composant TextInput	Champ d'entrée de texte à une ligne modifiable.
Composant Tree	Permet à un utilisateur de manipuler des informations hiérarchiques.
Composant Window	Fenêtre contenant une barre de titre, une légende, une bordure et un bouton Fermer et présentant du contenu à l'utilisateur.

Composants de support

Composant	Description
Composant MediaController	Contrôle la lecture de support en flux continu dans une application.
Composant MediaDisplay	Affiche le support en flux continu dans une application.
Composant MediaPlayback	Combinaison des composants MediaDisplay et MediaController.

Composants de données

Composant	Description
Paquet DataBinding	Ces classes implémentent la fonctionnalité de liaison des données Flash lors de l'exécution.
Composant DataHolder	Contient des données et peut être utilisé en tant que connecteur entre composants.
Composant DataProvider	Ce composant est le modèle des listes de données à accès linéaire. Ce modèle offre des capacités simples de manipulation des tableaux qui diffusent leurs modifications.
Composant DataSet	Bloc de construction pour la création d'applications de données.
Composant RDBMSResolver	Permet d'enregistrer les données sur n'importe quelle source de données supportée. Ce composant Resolver traduit le format XML qui peut être reçu et analysé par un service web, JavaBean, servlet ou une page ASP.
Composant WebServiceConnector	Fournit un accès sans script aux appels de méthode de service web.
Composant XMLConnector	Lit et rédige des documents XML à l'aide des méthodes HTTP GET et POST.
Composant XUpdateResolver	Permet d'enregistrer les données sur n'importe quelle source de données supportée. Ce composant Resolver traduit le paquet Delta au format XUpdate.

Gestionnaires

Composant	Description
Classe DepthManager	Gère la profondeur des objets dans les piles.
Classe FocusManager	Gère la navigation entre les composants à l'écran à l'aide de la touche de tabulation. Gère également les changements de focus lorsque les utilisateurs cliquent dans l'application.
Classe PopUpManager	Permet de créer et de supprimer des fenêtres contextuelles.
Classe StyleManager	Vous permet d'enregistrer les styles et de gérer les styles hérités.

Ecrans

Composant	Description
Classe Slide	Permet de manipuler les écrans de présentation de diapositives lors de l'exécution.
Classe Form	Permet de manipuler les écrans d'applications de formulaires lors de l'exécution.

Composant Accordion

Pour obtenir les informations les plus récentes sur cette fonction, cliquez sur le bouton Mettre à jour dans la partie supérieure de l'onglet Aide.

Composant Alert

Pour obtenir les informations les plus récentes sur cette fonction, cliquez sur le bouton Mettre à jour dans la partie supérieure de l'onglet Aide.

Composant Button

Le composant Button est un bouton de l'interface utilisateur rectangulaire dont les dimensions peuvent être modifiées. Vous pouvez ajouter une icône personnalisée à un bouton. Vous pouvez également modifier son comportement pour le faire passer de la pression au basculement. Un bouton à basculement reste enfoncé une fois que vous avez cliqué dessus et retourne à son état relevé lorsque vous cliquez de nouveau dessus.

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

Touche	Description
Maj +Tab	Place le focus sur l'objet précédent.
Espace	Active ou Désactive le composant et déclenche l'événement `click`.
Tab	Place le focus sur l'objet suivant.

Pour plus d'informations sur le contrôle du focus, consultez *Création de la navigation personnalisée du focus*, page 25 ou *Classe FocusManager*, page 103.

Un aperçu direct des occurrences de bouton reflète les modifications apportées aux paramètres dans l'inspecteur des propriétés ou le panneau Inspecteur de composants pendant la création. Cependant, dans l'aperçu en direct, une icône personnalisée est représentée sur la scène par un carré gris.

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

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

Vous n'activez l'accessibilité d'un composant qu'une fois, quel que soit son nombre d'occurrences. Pour plus d'informations, consultez « Création de contenu accessible », dans le guide Utilisation de Flash de l'aide. Il vous faudra parfois mettre votre système d'aide à jour pour obtenir ces informations.

Utilisation du composant Button

Les boutons constituent des éléments fondamentaux de toutes les formes d'application web. Vous pouvez utiliser des boutons partout où vous souhaitez que les utilisateurs lancent un événement. Par exemple, la plupart des formulaires comportent un bouton « Envoyer ». Vous pouvez également ajouter des boutons « Précédent » et « Suivant » à une présentation.

Pour ajouter une icône à un bouton, vous devez sélectionner ou créer un clip ou un symbole graphique que vous utiliserez comme icône. Le symbole doit être enregistré sous 0, 0, pour une mise en forme appropriée sur le bouton. Choisissez le symbole de l'icône dans le panneau Bibliothèque, ouvrez la boîte de dialogue Liaison dans le menu d'options et saisissez un identificateur de liaison. Il s'agit de la valeur à entrer pour le paramètre de l'icône dans l'inspecteur des propriétés ou le panneau Inspecteur de composants. Vous pouvez également entrer cette valeur pour la propriété Button.icon d'ActionScript.

Remarque : Si la taille d'une icône est supérieure au bouton, elle s'étend au-delà des bordures du bouton.

Paramètres du bouton

Voici les paramètres de création à définir pour chaque occurrence du composant Button dans l'inspecteur des propriétés ou dans le panneau Inspecteur de composants :

label définit la valeur du texte inscrit sur le bouton ; la valeur par défaut est Bouton.

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

toggle transforme le bouton en bouton à basculement. Si la valeur est vraie (true), le bouton reste dans l'état enfoncé lorsque l'on appuie dessus et retourne à l'état relevé lorsque l'on appuie de nouveau dessus. Si la valeur est fausse (false) , le bouton se comporte comme un bouton-poussoir normal ; la valeur par défaut est fausse.

selected si le paramètre de basculement est vrai, ce paramètre spécifie si le bouton est enfoncé (vrai) ou relâché (faux). La valeur par défaut est false.

labelPlacement oriente le texte de l'étiquette sur le bouton par rapport à l'icône. Ce paramètre peut avoir l'un des quatre paramètres suivants : gauche, droite, haut ou bas ; la valeur par défaut est droite. Pour plus d'informations, consultez `Button.labelPlacement`.

Vous pouvez rédiger des instructions ActionScript pour contrôler ces paramètres ainsi que d'autres options pour les composants Button en vous servant de leurs propriétés, de leurs méthodes et de leurs événements. Pour plus d'informations, consultez Classe Button.

Création d'une application avec le composant Button

La procédure suivante explique comment ajouter un composant Button à une application en mode de création. Dans cet exemple, le bouton est un bouton Aide doté d'une icône personnalisée qui permet d'accéder à un système d'aide lorsque l'utilisateur appuie dessus.

Pour créer une application avec le composant Button, procédez comme suit :

1 Faites glisser un composant Button du panneau Composants vers la scène.

2 Dans l'inspecteur des propriétés, entrez **BtnAide** comme nom d'occurrence.

3 Dans l'inspecteur des propriétés, procédez comme suit :

- Entrez **Help** comme paramètre de l'étiquette.
- Entrez **HelpIcon** comme paramètre de l'icône.

  Pour utiliser une icône, un clip ou un symbole graphique doit être stocké dans la bibliothèque avec un identificateur de liaison, à utiliser comme paramètre de l'icône. Dans cet exemple, l'identificateur de liaison est HelpIcon.
- Définissez la propriété de basculement sur vrai.

4 Choisissez l'image 1 dans le scénario, ouvrez le panneau Actions et saisissez le code suivant :

```
clippyListener = new Object();
clippyListener.click = function (evt){
  clippyHelper.enabled = evt.target.selected;
}
BtnAide.addEventListener("click", clippyListener);
```

La dernière ligne de code ajoute un gestionnaire d'événements `click` à l'occurrence `BtnAide`. Le gestionnaire active et désactive l'occurrence `clippyHelper` susceptible de servir de panneau Aide.

Personnalisation du composant Button

Vous pouvez orienter un composant Button dans le sens horizontal et vertical pendant la création et l'exécution. Lors de la création, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. Lors de l'exécution, utilisez la méthode `setSize()` (consultez `UIObject.setSize()`) ou des propriétés et méthodes applicables de la classe Button (consultez Classe Button). Le redimensionnement du bouton n'affecte pas la taille de l'icône ou de l'étiquette.

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

Si la taille d'une icône est supérieure au bouton, elle s'étend au-delà des bordures du bouton.

Utilisation de styles avec le composant Button

Vous pouvez définir des propriétés de style afin de modifier l'aspect de l'occurrence d'un bouton. Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, consultez *Utilisation des styles pour personnaliser la couleur et le texte des composants*, page 27.

Les composants Button supportent les styles de halo suivants :

Style	Description
`themeColor`	Arrière-plan d'un composant. Il s'agit du seul style de couleur qui n'hérite pas de sa valeur. "haloGreen", "haloBlue" et "haloOrange" sont des valeurs possibles.
`color`	Texte d'une étiquette de composant.
`disabledColor`	Couleur désactivée pour du texte.
`fontFamily`	Nom de police pour du texte.
`fontSize`	Taille en points pour la police.
`fontStyle`	Style de police : "normal" ou "italic".
`fontWeight`	Epaisseur de la police : "normal" ou "bold".

Utilisation des enveloppes avec le composant Button

Le composant Button utilise l'API de dessin ActionScript pour dessiner les états des boutons. Pour envelopper un composant Button au cours de la programmation, modifiez le code ActionScript qui se trouve dans le fichier ButtonSkin.as situé dans le dossier First Run\Classes\mx\skins\halo.

Si vous utilisez la méthode `UIObject.createClassObject()` pour créer dynamiquement une occurrence de composant Button (pendant l'exécution), vous pouvez lui appliquer une enveloppe dynamiquement. Pour appliquer un composant lors de l'exécution, définissez les propriétés d'enveloppe du paramètre `initObject` qui est passé à la méthode `createClassObject()`. Ces propriétés d'enveloppe définissent les noms des symboles à utiliser en tant qu'états du bouton, avec et sans icône.

Si vous définissez le paramètre de l'icône au moment de la programmation ou la propriété `icon` d'ActionScript à l'exécution, le même identificateur de liaison est affecté aux trois états de l'icône : `falseUpIcon`, `falseDownIcon` et `trueUpIcon`. Pour désigner une icône unique pour les huit états d'icône (si par exemple vous voulez qu'une icône différente apparaisse lorsqu'un utilisateur appuie sur un bouton), vous devez définir les propriétés du paramètre `initObject` utilisé dans la méthode `createClassObject()`.

Le code suivant crée un objet intitulé `initObject`, à utiliser en tant que paramètre `initObject`, et définit les propriétés de l'enveloppe sur les nouveaux identificateurs de liaison du symbole. La dernière ligne de code appelle la méthode `createClassObject()` pour créer une nouvelle occurrence de la classe Button avec les propriétés du paramètre `initObject`, comme suit :

```
var initObject = new Object();
initObject.falseUpIcon = "MyFalseUpIcon";
initObject.falseDownIcon = "MyFalseDownIcon";
initObject.trueUpIcon = "MytrueUpIcon";
createClassObject(mx.controls.Button, "ButtonInstance", 0, initObject);
```

Pour plus d'informations, consultez *A propos de l'application des enveloppes aux composants*, page 37 et `UIObject.createClassObject()`.

Si un bouton est activé, il affiche son état survolé lorsque le pointeur passe dessus. Le bouton reçoit le focus d'entrée et affiche son état enfoncé lorsque l'on clique dessus. Le bouton retourne à son état survolé lorsque la souris est relâchée. Si le pointeur quitte le bouton alors que le bouton de la souris est enfoncé, le bouton retourne à son état d'origine et garde le focus d'entrée. Si le paramètre de basculement est défini sur vrai, l'état du bouton ne change pas jusqu'à ce que le bouton de la souris soit relâché sur lui.

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

Les composants Button utilisent les propriétés d'enveloppe suivantes :

Propriété	Description
`falseUpSkin`	Etat En haut. La valeur par défaut est RectBorder.
`falseDownSkin`	Etat Enfoncé. La valeur par défaut est RectBorder.
`falseOverSkin`	Etat Survolé. La valeur par défaut est RectBorder.
`falseDisabledSkin`	Etat Désactivé. La valeur par défaut est RectBorder.
`trueUpSkin`	Etat Basculé. La valeur par défaut est RectBorder.
`trueDownSkin`	Etat Enfoncé-basculé. La valeur par défaut est RectBorder.
`trueOverSkin`	Etat Survolé-basculé. La valeur par défaut est RectBorder.
`trueDisabledSkin`	Etat Désactivé-basculé. La valeur par défaut est RectBorder.
`falseUpIcon`	Etat En haut de l'icône. La valeur par défaut est undefined.
`falseDownIcon`	Etat Enfoncé de l'icône. La valeur par défaut est undefined.
`falseOverIcon`	Etat Survolé de l'icône. La valeur par défaut est undefined.
`falseDisabledIcon`	Etat Désactivé de l'icône. La valeur par défaut est undefined.
`trueUpIcon`	Etat Basculé de l'icône. La valeur par défaut est undefined.
`trueOverIcon`	Etat Survolé-basculé de l'icône. La valeur par défaut est undefined.
`trueDownIcon`	Etat Enfoncé-basculé de l'icône. La valeur par défaut est undefined.
`trueDisabledIcon`	Etat Désactivé-basculé de l'icône. La valeur par défaut est undefined.

Classe Button

Héritage UIObject > UIComponent > SimpleButton > Button

Espace de nom de classe ActionScript mx.controls.Button

Les propriétés de la classe Button vous permettent d'ajouter une icône à un bouton, de créer une étiquette de texte ou d'indiquer si le bouton agit en tant que bouton-poussoir ou en tant que bouton à basculement pendant l'exécution.

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

Le composant Button utilise FocusManager pour remplacer le rectangle de focus par défaut de Flash Player et tracer un rectangle de focus personnalisé aux coins arrondis. Pour plus d'informations, consultez *Création de la navigation personnalisée du focus*, page 25.

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

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

Remarque : Le code suivant renvoie la valeur undefined : `trace(OccurrenceMonBouton.version);`.

La classe de composants Button est différente de l'objet Button ActionScript intégré.

Résumé des méthodes de la classe Button

Hérite de toutes les méthodes de la Classe UIObject et de la Classe UIComponent.

Résumé des propriétés de la classe Button

Méthode	Description
`SimpleButton.emphasized`	Indique si un bouton a l'aspect d'un bouton-poussoir par défaut.
`SimpleButton.emphasizedStyleDeclaration`	Déclaration de style lorsque la propriété `emphasized` est définie sur `true`.
`Button.icon`	Spécifie une icône pour une occurrence de bouton.
`Button.label`	Spécifie le texte qui apparaît dans un bouton.
`Button.labelPlacement`	Spécifie l'orientation du texte de l'étiquette par rapport à une icône.
`Button.selected`	Lorsque la propriété `toggle` est `true`, spécifie si le bouton est enfoncé (`true`) ou non (`false`).
`Button.toggle`	Indique si le bouton se comporte comme un bouton à basculement.

Hérite de toutes les propriétés de Classe UIObject et de la Classe UIComponent.

Résumé des événements de la classe Button

Méthode	Description
`Button.click`	Diffusé lorsque l'utilisateur clique sur le bouton de la souris au-dessus d'une occurrence de bouton ou appuie sur la barre d'espace.

Hérite de tous les événements de la Classe UIObject et de la Classe UIComponent.

Button.click

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

```
ObjetDécoute = new Object();
ObjetDécoute.click = function(eventObject){
   ...
}
buttonInstance.addEventListener("click", listenerObject)
```

Description

Evénement : diffuse à tous les écouteurs enregistrés lorsque l'utilisateur clique sur le bouton (puis relâche le bouton de la souris) ou si le bouton a le focus et que l'utilisateur appuie sur la barre d'espacement.

Le premier exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence de composant Button. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de comportement. Par exemple, le code suivant, joint à l'occurrence de composant Button `monComposantBouton`, envoie « _level0.monComposantBouton » au panneau de sortie :

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

Notez que ce code est différent du comportement de `this` lorsqu'il est utilisé dans un gestionnaire `on()` lié à un symbole de bouton Flash ordinaire. Lorsque `this` est utilisé à l'intérieur d'un gestionnaire `on()` lié à un symbole de bouton, il fait référence au scénario contenant le bouton. Par exemple, le code suivant, lié à l'occurrence de symbole de bouton `monBouton`, envoie « _level0 » au panneau de sortie :

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

Remarque : L'objet bouton ActionScript intégré n'a pas d'événement `click` ; l'événement le plus proche est `release`.

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`occurrenceDeBouton`*) distribue un événement (ici, `click`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. L'objet événement a un jeu de propriétés contenant des informations sur l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `addEventListener()` (consultez `UIEventDispatcher.addEventListener()`) sur l'occurrence de composant qui distribue l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

Cet exemple, écrit sur une image du scénario, envoie un message au panneau de sortie lorsque l'utilisateur clique sur un bouton intitulé `occurrenceDeBouton`. La première ligne de code donne une étiquette au bouton. La deuxième ligne lui applique un comportement de bouton à basculement. La troisième ligne crée un objet d'écoute intitulé `form`. La quatrième ligne définit une fonction pour l'événement `click` sur l'objet d'écoute. La fonction comporte une action trace qui utilise l'objet événement automatiquement transmis à cette fonction (ici `objEvnt`) pour générer un message. La propriété `target` d'un objet événement est le composant ayant généré l'événement (dans cet exemple, `occurrenceDeBouton`). L'utilisateur accède à la propriété `Button.selected` à partir de la propriété `target` de l'objet événement. La dernière ligne appelle la méthode `addEventListener()` à partir de `occurrenceDeBouton` et lui passe l'événement `click` et l'objet d'écoute `form` comme paramètres, comme dans le code suivant :

```
occurrenceDeBouton.label = "Cliquez sur Tester"
occurrenceDeBouton.toggle = true;
form = new Object();
form.click = function(objEvnt){
  trace("La propriété sélectionnée est passée à " + objEvnt.target.selected);
}
occurrenceDeBouton.addEventListener("click", form);
```

Le code suivant envoie également un message au panneau de sortie lorsque l'utilisateur clique sur `occurrenceDeBouton`. Le gestionnaire `on()` doit être directement lié à `occurrenceDeBouton`, comme dans l'exemple ci-dessous :

```
on(click){
  trace("composant Button cliqué");
}
```

Consultez également

`UIEventDispatcher.addEventListener()`

SimpleButton.emphasized

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004

Usage

`occurrenceDeBouton``.emphasized`

Description

Propriété : indique si le bouton est dans un état emphasized (`true`) ou non (`false`). L'état emphasized correspond à l'aspect s'il s'agit d'un bouton-poussoir par défaut. En général, utilisez la propriété `FocusManager.defaultPushButton` au lieu de définir la propriété `emphasized` directement. La valeur par défaut est `false`.

La propriété emphasized est une propriété statique de la classe SimpleButton. Vous devez donc y accéder directement à partir de SimpleButton, de la manière suivante :

```
SimpleButton.emphasizedStyleDeclaration = "foo";
```

Si vous n'utilisez pas `FocusManager.defaultPushButton`, il est possible que vous souhaitiez simplement définir un bouton sur l'état emphasized ou utiliser l'état emphasized pour choisir une autre couleur pour le texte. L'exemple suivant définit la propriété `emphasized` pour l'occurrence de bouton `monBouton` :

```
_global.styles.foo = new CSSStyleDeclaration();
_global.styles.foo.color = 0xFF0000;
SimpleButton.emphasizedStyleDeclaration = "foo";
monBouton.emphasized = true;
```

Consultez également

`SimpleButton.emphasizedStyleDeclaration`

SimpleButton.emphasizedStyleDeclaration

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004

Usage

occurrenceDeBouton`.emphasizedStyleDeclaration`

Description

Propriété : une chaîne indiquant la déclaration de style qui formate un bouton lorsque la propriété `emphasized` est définie sur `true`.

Consultez également

`Window.titleStyleDeclaration`, `SimpleButton.emphasized`

Button.icon

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceDeBouton`.icon`

Description

Propriété : une chaîne spécifiant l'identificateur de liaison d'un symbole dans la bibliothèque. Cet identificateur sera utilisé comme icône pour l'occurrence d'un bouton. L'icône peut être un symbole de clip ou un symbole graphique avec un point d'alignement supérieur gauche. Vous devez redimensionner le bouton si l'icône est trop grande ; le bouton et l'icône ne seront pas redimensionnés automatiquement. Si une icône est plus grande qu'un bouton, elle s'étend automatiquement au-delà des bordures du bouton.

Pour créer une icône personnalisée, vous devez créer un clip ou un symbole graphique. Sélectionnez le symbole sur la scène en mode de modification des symboles et saisissez 0 dans les cases X et Y de l'inspecteur des propriétés. Dans le panneau Bibliothèque, sélectionnez le clip et choisissez Liaison dans le menu d'options. Sélectionnez Exporter pour ActionScript et saisissez un identificateur dans la case Identifiant.

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

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

Exemple

Le code suivant affecte le clip ayant l'identificateur de liaison `bonheur` dans le panneau Bibliothèque à l'occurrence de bouton en tant qu'icône :

```
monBouton.icon = "bonheur"
```

Consultez également

`Button.labelPlacement`

Button.label

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`occurrenceDeBouton``.label`

Description

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

Exemple

Le code suivant définit l'étiquette sur « Supprimer de la liste » :

```
occurrenceDeBouton.label = "Supprimer de la liste";
```

Consultez également

`Button.labelPlacement`

Button.labelPlacement

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceDeBouton.labelPlacement

Description

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

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

Exemple

Le code suivant définit l'étiquette à gauche du bouton. La deuxième ligne de code envoie la valeur de la propriété `labelPlacement` au panneau de sortie :

```
occurrenceDicône.labelPlacement = "left";
trace(occurrenceDicône.labelPlacement);
```

Button.selected

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceDeBouton.selected

Description

Propriété : valeur booléenne indiquant qu'un bouton est enfoncé (`true`) ou non (`false`). La valeur de la propriété `toggle` doit être `true` (vraie) pour définir la propriété `selected` sur `true` (vraie). Si la propriété `toggle` est `false`, l'affectation d'une valeur `true` sur la propriété `selected` n'a aucun effet. La valeur par défaut est `false`.

L'événement `click` n'est pas déclenché lorsque la valeur de la propriété `selected` change avec ActionScript. Il est déclenché lorsqu'un utilisateur établit une interaction avec le bouton.

Exemple

Dans l'exemple suivant, la propriété `toggle` est définie sur `true` et la propriété `selected` est également définie sur `true`, ce qui fait passer le bouton à l'état enfoncé. L'action `trace` envoie la valeur `true` au panneau de sortie :

```
occurrenceDeBouton.toggle = true; // toggle doit être true pour définir la
  propriété selected
occurrenceDeBouton.selected = true; // affiche l'état basculé du bouton
trace(occurrenceDeBouton.selected); //trace- true
```

Consultez également

`Button.toggle`

Button.toggle

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`occurrenceDeBouton``.toggle`

Description

Propriété : une valeur booléenne indiquant si un bouton agit comme un bouton à basculement (`true`) ou comme un bouton-poussoir (`false`) ; la valeur par défaut est `false`. Lorsqu'un bouton à basculement est enfoncé, il reste dans l'état enfoncé jusqu'à ce que l'on clique de nouveau dessus.

Exemple

Le code suivant définit la propriété `toggle` sur `true` pour que l'occurrence `monBouton` se comporte comme un bouton à basculement :

```
monBouton.toggle = true;
```

Interface CellRenderer

Pour des informations actualisées sur cette fonction, cliquez sur le bouton Mettre à jour, dans la partie supérieure de l'onglet Aide.

Composant CheckBox

Une case à cocher est une case carrée pouvant être sélectionnée ou désélectionnée. Lorsqu'elle est sélectionnée, une coche apparaît à l'intérieur. Vous pouvez ajouter une étiquette de texte à une case à cocher et la placer à gauche, à droite, au-dessous ou au-dessus.

Les cases peuvent être activées ou désactivées dans une application. Si une case est activée et qu'un utilisateur clique dessus ou sur son étiquette, la case reçoit le focus d'entrée et son état enfoncé apparaît à l'écran. Si un utilisateur place le pointeur à l'extérieur du cadre de délimitation d'une case ou de son étiquette en appuyant sur le bouton de la souris, l'aspect du composant revient à son état d'origine et conserve le focus d'entrée. L'état d'une case ne change pas jusqu'à ce que le bouton de la souris soit relâché sur le composant. La case possède aussi deux états désactivés, sélectionné ou désélectionné, qui ne permettent pas d'établir une interaction avec la souris ou le clavier.

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

Une occurrence de case à cocher reçoit le focus si un utilisateur clique dessus ou utilise la touche de tabulation pour la sélectionner. Lorsqu'une occurrence de case a le focus, vous pouvez utiliser les touches suivantes pour le contrôler :

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

Pour plus d'informations sur le contrôle du focus, consultez *Création de la navigation personnalisée du focus*, page 25 ou *Classe FocusManager*, page 103.

Un aperçu en direct de chacune des occurrences de case reflète les modifications apportées à leurs paramètres dans le panneau de l'inspecteur des propriétés ou des composants au cours de la programmation.

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

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

Vous n'activez l'accessibilité d'un composant qu'une fois, quel que soit son nombre d'occurrences. Pour plus d'informations, consultez « Création de contenu accessible », dans le guide Utilisation de Flash de l'aide. Il vous faudra parfois mettre votre système d'aide à jour pour obtenir ces informations.

Utilisation du composant CheckBox

La case à cocher est l'un des éléments fondamentaux des formulaires et applications web. Vous pouvez utiliser des cases chaque fois que vous voulez réunir un jeu de valeurs `true` ou `false` qui ne s'excluent pas réciproquement. Par exemple, un formulaire recueillant des informations personnelles sur un client peut comporter une liste de hobbies que le client doit sélectionner ; une case à cocher peut se trouver à côté de chaque loisir.

Paramètres du composant CheckBox

Les paramètres suivants sont des paramètres de programmation que vous pouvez définir pour chaque occurrence de composant CheckBox dans le panneau de l'inspecteur des propriétés ou des composants :

label définit la valeur du texte sur la case à cocher ; la valeur par défaut est defaultValue.

selected définit la valeur initiale de la case cochée (true) ou non cochée (false).

labelPlacement oriente le texte de l'étiquette sur la case. Ce paramètre peut avoir l'un des quatre paramètres suivants : gauche, droite, haut ou bas ; la valeur par défaut est droite. Pour plus d'informations, consultez `CheckBox.labelPlacement`.

Vous pouvez rédiger du code ActionScript pour contrôler ces options ainsi que d'autres options des composants CheckBox à l'aide des propriétés, méthodes et événements ActionScript. Pour plus d'informations, consultez Classe CheckBox.

Création d'une application avec le composant CheckBox

La procédure suivante explique comment ajouter un composant CheckBox à une application au cours de la programmation. L'exemple suivant est un formulaire à remplir pour s'inscrire dans une agence matrimoniale en ligne. Le formulaire est une requête qui recherche des candidats susceptibles de convenir au client. Le formulaire de requête doit comporter une case intitulée « Limiter l'âge » pour permettre au client de limiter sa recherche au groupe d'âge de son choix. Lorsque la case « Limiter l'âge » est cochée, le client peut alors entrer un âge minimum et un âge maximum dans les deux champs de texte qui sont uniquement activés lorsque la case « Limiter l'âge » est sélectionnée.

Pour créer une application avec le composant CheckBox, procédez comme suit :

1. Faites glisser deux composants TextInput du panneau Composants jusqu'à la scène.
2. Dans l'inspecteur des propriétés, saisissez les noms d'occurrence `âgeMinimum` et `âgeMaximum`.
3. Faites glisser un composant CheckBox du panneau Composants jusqu'à la scène.
4. Dans l'inspecteur des propriétés, procédez comme suit :
   - Saisissez **restrictAge** pour le nom de l'occurrence.
   - Saisissez **Limiter l'âge** pour le paramètre de l'étiquette.
5. Choisissez l'image 1 dans le scénario, ouvrez le panneau Actions et saisissez le code suivant :

```
restrictAgeListener = new Object();
restrictAgeListener.click = function (evt){
  âgeMinimum.enabled = evt.target.selected;
  âgeMaximum.enabled = evt.target.selected;
}
restrictAge.addEventListener("click", restrictAgeListener);
```

Ce code crée un gestionnaire d'événements `click` qui active et désactive les composants des champs de texte `âgeMinimum` et `âgeMaximum`, déjà placés sur la scène. Pour plus d'informations sur l'événement `click`, consultez `CheckBox.click`. Pour plus d'informations sur le composant TextInput , consultez *Composant TextInput*, page 220.

Personnalisation du composant CheckBox

Vous pouvez transformer un composant CheckBox horizontalement et verticalement au cours de la programmation et lors de l'exécution. Lors de la création, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. Lors de l'exécution, utilisez la méthode `setSize()` (`UIObject.setSize()`) ou les autres propriétés et méthodes de la classe CheckBox (consultez Classe CheckBox). Le redimensionnement de la case ne modifie pas la taille de l'étiquette ou de l'icône ; il modifie uniquement la taille du cadre de délimitation.

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

Utilisation des styles avec le composant CheckBox

Vous pouvez définir des propriétés de style pour modifier l'aspect d'une occurrence de case à cocher. Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, consultez *Utilisation des styles pour personnaliser la couleur et le texte des composants*, page 27.

Les composants CheckBox supportent les styles de halo suivants :

Style	Description
`themeColor`	Arrière-plan d'un composant. Il s'agit du seul style de couleur qui n'hérite pas de sa valeur. "haloGreen", "haloBlue" et "haloOrange" sont des valeurs possibles.
`color`	Texte d'une étiquette de composant.
`disabledColor`	Couleur désactivée pour du texte.
`fontFamily`	Nom de police pour du texte.
`fontSize`	Taille en points pour la police.
`fontStyle`	Style de police : "normal" ou "italic".
`fontWeight`	Epaisseur de la police : "normal" ou "bold".
`textDecoration`	Décoration du texte : "none" ou "underline".

Utilisation des enveloppes avec le composant CheckBox

Le composant CheckBox utilise les symboles du panneau Bibliothèque pour représenter les états des boutons. Pour envelopper le composant CheckBox au cours de la programmation, modifiez les symboles dans le panneau Bibliothèque. Les enveloppes de composants CheckBox sont situées dans le dossier Flash UI Components 2/Themes/MMDefault/CheckBox Assets/states de la bibliothèque du fichier HaloTheme.fla ou du fichier SampleTheme.fla. Pour plus d'informations, consultez *A propos de l'application des enveloppes aux composants*, page 37.

Les composants CheckBox utilisent les propriétés d'enveloppe suivantes :

Propriété	Description
`falseUpSkin`	Etat En haut. La propriété par défaut est `RectBorder`.
`falseDownSkin`	Etat Enfoncé. La propriété par défaut est `RectBorder`.
`falseOverSkin`	Etat Survolé. La propriété par défaut est `RectBorder`.
`falseDisabledSkin`	Etat Désactivé. La propriété par défaut est `RectBorder`.
`trueUpSkin`	Etat Basculé. La propriété par défaut est `RectBorder`.
`trueDownSkin`	Etat Enfoncé-basculé. La propriété par défaut est `RectBorder`.
`trueOverSkin`	Etat Survolé-basculé. La propriété par défaut est `RectBorder`.
`trueDisabledSkin`	Etat Désactivé-basculé. La propriété par défaut est `RectBorder`.

Classe CheckBox

Héritage UIObject > UIComponent > SimpleButton > Button > CheckBox

Espace de nom de classe ActionScript mx.controls.CheckBox

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

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

Le composant CheckBox utilise FocusManager pour remplacer le rectangle de focus de Flash Player par défaut et tracer un rectangle de focus personnalisé avec des coins arrondis. Pour plus d'informations, consultez *Création de la navigation personnalisée du focus*, page 25.

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

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

Remarque : Le code suivant renvoie la valeur undefined : `trace(monOccurrenceDeCaseACocher.version);`.

Résumé des propriétés pour la classe CheckBox

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

Hérite de toutes les propriétés de Classe UIObject et Classe UIComponent.

Méthodes de la classe CheckBox

Hérite de toutes les méthodes de la Classe UIObject et de la Classe UIComponent.

Résumé des événements pour la classe CheckBox

Evénement	Description
CheckBox.click	Déclenché lorsque l'on appuie sur le bouton de la souris au-dessus d'une occurrence de bouton.

Hérite de tous les événements de la Classe UIObject, et de la Classe UIComponent.

CheckBox.click

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

```
listenerObject = new Object();
listenerObject.click = function(eventObject){
  ...
}
OccurrenceDeCaseACocher.addEventListener("click", objetDécoute)
```

Description

Evénement : diffuse à tous les écouteurs enregistrés lorsque la souris est enfoncée (relâchée) sur le bouton ou si le bouton a le focus et que la barre d'espacement est enfoncée.

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

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*OccurrenceDeCaseACocher*) distribue un événement (ici, `click`) qui est géré par une fonction associée à un objet d'écoute (*objetDécoute*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*objetEvnt*) à la méthode d'objet d'écoute. L'objet événement a un jeu de propriétés contenant des informations sur l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `addEventListener()` (consultez `UIEventDispatcher.addEventListener()`) sur l'occurrence de composant qui diffuse l'événement afin d'enregistrer l'écouteur dans cette occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

Cet exemple, écrit sur une image du scénario, envoie un message au panneau de sortie lorsque l'utilisateur clique sur un bouton intitulé `OccurrenceDeCaseACocher`. La première ligne de code crée un objet d'écoute intitulé `form`. La deuxième ligne définit une fonction pour l'événement `click` sur l'objet d'écoute. La fonction comporte une action `trace` qui utilise l'objet événement (ici `objEvnt)` qui lui est automatiquement transmis pour générer un message. La propriété `target` d'un objet événement est le composant ayant généré l'événement (dans cet exemple, `OccurrenceDeCaseACocher`). L'utilisateur accède à la propriété `CheckBox.selected` à partir de la propriété `target` de l'objet événement. La dernière ligne appelle la méthode `addEventListener()` à partir de `OccurrenceDeCaseACocher` et lui transmet l'événement `click` et l'objet d'écoute `form` comme paramètres, comme dans l'exemple suivant :

```
form = new Object();
form.click = function(objEvnt){
  trace("La propriété sélectionnée est passée à " + objEvnt.target.selected);
}
OccurrenceDeCaseACocher.addEventListener("click", form);
```

Le code suivant envoie également un message au panneau de sortie lorsque l'utilisateur clique sur `OccurrenceDeCaseACocher`. Le gestionnaire `on()` doit être directement lié à `OccurrenceDeCaseACocher`, comme dans l'exemple suivant :

```
on(click){
  trace("composant case à cocher cliqué");
}
```

Consultez également

`UIEventDispatcher.addEventListener()`

CheckBox.label

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeCaseACocher`.label`

Description

Propriété : spécifie l'étiquette de texte pour la case à cocher. Par défaut, l'étiquette apparaît à droite de la case à cocher. La définition de cette propriété remplace le paramètre d'étiquette spécifié dans le panneau des paramètres de clip.

Exemple

Le code suivant définit le texte qui apparaît à côté du composant CheckBox et envoie la valeur au panneau de sortie :

```
checkBox.label = "Supprimer de la liste";
trace(checkBox.label)
```

Consultez également

`CheckBox.labelPlacement`

CheckBox.labelPlacement

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`OccurrenceDeCaseACocher``.labelReplacement`

Description

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

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

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

- `"bottom"` L'étiquette est placée sous la case. Le regroupement de la case et de l'étiquette est centré horizontalement et verticalement.

- `"top"` L'étiquette est placée au-dessus de la case. Le regroupement de la case et de l'étiquette est centré horizontalement et verticalement.

Vous pouvez modifier le cadre de délimitation du composant au cours de la programmation à l'aide de la commande Transformer ou pendant l'exécution au moyen de la propriété `UIObject.setSize()`. Pour plus d'informations, consultez *Personnalisation du composant CheckBox*, page 64.

Exemple

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

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

Consultez également.

`CheckBox.label`

CheckBox.selected

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`OccurrenceDeCaseACocher``.selected`

Description

Propriété : une valeur booléenne qui sélectionne (`true`) ou désélectionne (`false`) la case.

Exemple

L'exemple suivant sélectionne l'occurrence `caseacocher1` :

```
caseacocher1.selected = true;
```

Composant ComboBox

Une liste déroulante peut être statique ou modifiable. Les listes déroulantes statiques permettent aux utilisateurs d'effectuer une sélection unique dans une liste déroulante. Les listes déroulantes modifiables permettent aux utilisateurs de saisir du texte directement dans une zone de texte, en haut de la liste, et de sélectionner un élément. Si la liste déroulante touche le bas du document, elle se déroule vers le haut au lieu de se dérouler vers le bas. Elle comporte trois sous-composants : Button, TextInput et List.

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

Un composant ComboBox reçoit le focus si vous cliquez sur la zone de texte ou sur le bouton. Lorsqu'un composant ComboBox a le focus et peut être modifié, toutes les frappes de touches vont dans la zone de texte et sont traitées selon les règles du composant TextInput (consultez *Composant TextInput*, page 220), à l'exception des touches suivantes :

Touche	Description
Contrôle+Bas	Ouvre la liste déroulante et lui donne le focus.
Maj +Tab	Place le focus sur l'objet précédent.
Tab	Place le focus sur l'objet suivant.

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

Touche	Description
Contrôle+Bas	Ouvre la liste déroulante et lui donne le focus.
Contrôle+Haut	Ferme la liste déroulante, le cas échéant.
Bas	La sélection se déplace d'un élément vers le bas.
Fin	La sélection se déplace jusqu'au bout de la liste.
Echap	Ferme la liste déroulante et place à nouveau le focus sur le composant ComboBox.
Entrée	Ferme la liste déroulante et place à nouveau le focus sur le composant ComboBox.
Origine	La sélection se déplace jusqu'au sommet de la liste.
Pg. Suiv.	La sélection se déplace sur la page suivante.
Pg. Préc.	La sélection se déplace sur la page précédente.
Maj +Tab	Place le focus sur l'objet précédent.
Tab	Place le focus sur l'objet suivant.

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

Touche	Description
Contrôle+Haut	Si la liste déroulante est ouverte, le focus retourne à la zone de texte et la liste se ferme.
Bas	La sélection se déplace d'un élément vers le bas.
Fin	Le point d'insertion est placé à la fin de la zone de texte.
Entrée	Si la liste déroulante est ouverte, le focus retourne à la zone de texte et la liste se ferme.

Touche	Description
Echap	Si la liste déroulante est ouverte, le focus retourne à la zone de texte et la liste se ferme.
Origine	Le point d'insertion est placé au début de la zone de texte.
Pg. Suiv.	La sélection se déplace sur la page suivante.
Pg. Préc.	La sélection se déplace sur la page précédente.
Tab	Place le focus sur l'objet suivant.
Maj-Fin	Sélectionne le texte compris entre le point d'insertion et la fin de la zone de texte.
Maj-Origine	Sélectionne le texte compris entre le point d'insertion et le début de la zone de texte.
Maj-Tab	Place le focus sur l'objet précédent.
Haut	La sélection se déplace d'un élément vers le haut.

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

Pour plus d'informations sur le contrôle du focus, consultez *Création de la navigation personnalisée du focus*, page 25 ou *Classe FocusManager*, page 103.

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

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

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

Vous n'activez l'accessibilité d'un composant qu'une fois, quel que soit son nombre d'occurrences. Pour plus d'informations, consultez « Création de contenu accessible », dans le guide Utilisation de Flash de l'aide. Il vous faudra parfois mettre votre système d'aide à jour pour obtenir ces informations.

Utilisation du composant ComboBox

Vous pouvez utiliser un composant ComboBox dans n'importe quel formulaire ou application nécessitant un choix unique dans une liste. Par exemple, vous pouvez fournir une liste déroulante de départements dans un formulaire où les clients devront saisir leur adresse. Vous pouvez utiliser une liste déroulante modifiable pour plusieurs scénarios complexes. Par exemple, dans une application d'itinéraire routier, vous pouvez utiliser une liste déroulante modifiable pour que l'utilisateur y saisisse ses adresses de départ et d'arrivée. La liste déroulante pourrait alors contenir des adresses déjà saisies.

Paramètres du composant ComboBox

Les paramètres suivants sont des paramètres de programmation que vous pouvez définir pour chaque occurrence de composant ComboBox dans le panneau de l'inspecteur des propriétés ou des composants :

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

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

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

rowCount définit le nombre maximum d'éléments pouvant être affichés simultanément sans utiliser une barre de défilement. La valeur par défaut est 5.

Vous pouvez rédiger des instructions ActionScript pour définir d'autres options pour les occurrences ComboBox à l'aide des méthodes, des propriétés et des événements de la classe ComboBox. Pour plus d'informations, consultez Classe ComboBox.

Création d'une application avec le composant ComboBox

La procédure suivante explique comment ajouter un composant ComboBox à une application au cours de la programmation. Dans cet exemple, la liste déroulante contient des villes.

Pour créer une application avec le composant ComboBox, procédez comme suit :

1 Faites glisser un composant ComboBox du panneau Composants jusqu'à la scène.

2 Sélectionnez l'outil Transformer et redimensionnez le composant sur la scène.

La liste déroulante peut uniquement être redimensionnée sur la scène au cours de la programmation. En général, on ne modifie que la largeur pour que les entrées soient correctement affichées dans la liste.

3 Sélectionnez la liste déroulante et, dans l'inspecteur des propriétés, saisissez le nom d'occurrence **comboBox**.

4 Dans le panneau de l'inspecteur des composants ou des propriétés, procédez comme suit :

- Saisissez Paris, Bordeaux et Lyon pour le paramètre de l'étiquette. Double-cliquez sur le champ du paramètre de l'étiquette afin d'ouvrir la boîte de dialogue Valeurs. Cliquez ensuite sur le signe plus pour ajouter des éléments.
- Saisissez MN.swf, OR.swf et NH.swf pour le paramètre des données.

  Il s'agit des fichiers SWF imaginaires qui peuvent par exemple être chargés lorsqu'un utilisateur sélectionne une ville dans la liste déroulante.

5 Choisissez l'image 1 dans le scénario, ouvrez le panneau Actions et saisissez le code suivant :

```
form = new Object();
form.change = function (evt){
  trace(evt.target.selectedItem.label);
}
ListeDéroulante.addEventListener("change", form);
```

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

Personnalisation du composant ComboBox

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

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

Dans les listes déroulantes modifiables, le bouton est la seule zone active ; la zone de texte ne l'est pas. Pour les listes déroulantes statiques, le bouton et la liste déroulante constituent la zone active.

Utilisation de styles avec le composant ComboBox

Vous pouvez définir des propriétés de style afin de modifier l'aspect d'un composant ComboBox. Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, consultez *Utilisation des styles pour personnaliser la couleur et le texte des composants*, page 27.

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

- Le bouton est une occurrence Button et en utilise les styles. (consultez *Utilisation de styles avec le composant Button*, page 52).
- Le texte est une occurrence TextInput et en utilise les styles. (consultez *Utilisation des styles avec le composant TextInput*, page 222).
- La liste est une occurrence List et en utilise les styles. (consultez *Utilisation des styles avec le composant List*, page 117).

Les composants ComboBox utilisent les styles Halo suivants :

Style	Description
`themeColor`	Arrière-plan d'un composant. Il s'agit du seul style de couleur qui n'hérite pas de sa valeur. "haloGreen", "haloBlue" et "haloOrange" sont des valeurs possibles.
`color`	Texte d'une étiquette de composant.
`disabledColor`	Couleur désactivée pour du texte.
`fontFamily`	Nom de police pour du texte.
`fontSize`	Taille en points pour la police.
`fontStyle`	Style de police : "normal" ou "italic".
`fontWeight`	Epaisseur de la police : "normal" ou "bold".
`textDecoration`	Décoration du texte : "none" ou "underline".
`openDuration`	Le nombre de millisecondes nécessaires à l'ouverture de la liste déroulante. La valeur par défaut est 250.
`openEasing`	Référence pour une fonction d'interpolation qui contrôle l'animation de la liste déroulante. Par défaut sinus entrée/sortie. Pour d'autres équations, téléchargez une liste sur le site web de Robert Penner.

Utilisation d'enveloppes avec le composant ComboBox

Le composant ComboBox utilise les symboles du panneau Bibliothèque pour représenter les états des boutons. Il possède des variables d'enveloppe pour la flèche vers le bas. Il utilise aussi des enveloppes de barre de défilement et de liste. Pour appliquer une enveloppe au composant ComboBox au cours de la programmation, modifiez les symboles dans le panneau Bibliothèque et réexportez le composant en tant que fichier SWC. Les enveloppes du composant CheckBox sont situées dans le dossier Flash UI Components 2/Themes/MMDefault/ComboBox Assets/states de la bibliothèque du fichier HaloTheme.fla ou du fichier SampleTheme.fla. Pour plus d'informations, consultez *A propos de l'application des enveloppes aux composants*, page 37.

Les composants ComboBox utilisent les propriétés d'enveloppe suivantes :

Propriété	Description
ComboDownArrowDisabledName	Etat désactivé de la flèche bas. La propriété par défaut est `RectBorder`.
ComboDownArrowDownName	Etat Enfoncé de la flèche bas. La propriété par défaut est `RectBorder`.
ComboDownArrowUpName	Etat Relevé de la flèche bas. La propriété par défaut est `RectBorder`.
ComboDownArrowOverName	Etat Survolé de la flèche bas. La propriété par défaut est `RectBorder`.

Classe ComboBox

Héritage UIObject > UIComponent > ComboBase > ComboBox

Espace de nom de classe ActionScript mx.controls.ComboBox

Le composant ComboBox associe trois sous-composants séparés : Button, TextInput et List. La plupart des API des sous-composants sont disponibles directement à partir du composant ComboBox et sont répertoriées dans les tables Méthode, Propriété et Evénement de la classe ComboBox.

La liste du composant ComboBox est fournie sous la forme d'un tableau ou d'un objet DataProvider. Si vous utilisez un objet DataProvider, la liste change lors de l'exécution. La source des données ComboBox peut être modifiée dynamiquement en passant à un nouveau tableau ou objet DataProvider.

Les éléments d'une liste déroulante sont indexés par position, en commençant par le chiffre 0. Les éléments peuvent être les suivants :

- Un type de données de base.
- Un objet contenant une propriété `label` et une propriété `data`.

  Remarque : Un objet peut utiliser la propriété `ComboBox.labelFunction` ou `ComboBox.labelField` pour déterminer la propriété `label`.

Si l'élément est un type de données de base différent d'une chaîne, il est converti en chaîne. Si l'élément est un objet, la propriété `label` doit être une chaîne et la propriété `data` peut avoir n'importe quelle valeur ActionScript.

Les méthodes du composant ComboBox auxquelles vous fournissez des éléments ont deux paramètres, label et datas, qui se réfèrent aux propriétés ci-dessus. Les méthodes qui renvoient un élément le renvoient en tant qu'objet.

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

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

Remarque : Le code suivant renvoie la valeur undefined : `trace(monOccurrenceDeListeDéroulante.version);`.

Méthodes de la classe ComboBox

Propriété	Description
`ComboBox.addItem()`	Ajoute un élément à la fin de la liste.
`ComboBox.addItemAt()`	Ajoute un élément à la fin de la liste, à l'emplacement d'index spécifié.
`ComboBox.close()`	Ferme la liste déroulante.
`ComboBox.getItemAt()`	Renvoie l'élément à l'emplacement d'index spécifié.
`ComboBox.open()`	Ouvrez la liste déroulante.
`ComboBox.removeAll()`	Supprime tous les éléments de la liste.
`ComboBox.removeItemAt()`	Supprime un élément de la liste à l'emplacement spécifié.
`ComboBox.replaceItemAt()`	Remplace un élément de la liste par un autre élément spécifié.

Hérite de toutes les méthodes de la Classe UIObject et de la Classe UIComponent.

Propriétés de la classe ComboBox

Propriété	Description
`ComboBox.dataProvider`	Modèle de données pour les éléments de la liste.
`ComboBox.dropdown`	Renvoie une référence au composant List contenu dans ComboBox.
`ComboBox.dropdownWidth`	La largeur de la liste déroulante, en pixels.
`ComboBox.editable`	Indique si un composant ComboBox est modifiable ou non.
`ComboBox.labelField`	Indique le champ de données à utiliser en tant qu'étiquette pour la liste déroulante.
`ComboBox.labelFunction`	Spécifie une fonction pour calculer le champ de l'étiquette pour la liste déroulante.
`ComboBox.length`	Lecture seule. La longueur de la liste déroulante.
`ComboBox.rowCount`	Le nombre maximal d'éléments de la liste à afficher au même moment.
`ComboBox.selectedIndex`	L'index de l'élément sélectionné dans la liste déroulante.

Propriété	Description
ComboBox.selectedItem	La valeur de l'élément sélectionné dans la liste déroulante.
ComboBox.text	La chaîne de texte dans la zone de texte.
ComboBox.textField	Référence au composant TextInput dans la liste déroulante.
ComboBox.value	La valeur de la zone de texte (modifiable) ou de la liste déroulante (statique).

Hérite de toutes les propriétés de Classe UIObject et Classe UIComponent.

Evénements de la classe ComboBox

Evénement	Description
ComboBox.change	Diffusion lorsque la valeur de la liste déroulante change suite à l'interaction d'un utilisateur.
ComboBox.close	Diffusion lorsque la liste déroulante commence à se fermer.
ComboBox.enter	Diffusion lorsque la touche Entrée est enfoncée.
ComboBox.itemRollOut	Diffusion lorsque le pointeur survole un élément dans une liste déroulante.
ComboBox.itemRollOver	Diffusion lorsqu'un élément de liste déroulante est survolé.
ComboBox.open	Diffusion lorsque la liste déroulante commence à s'ouvrir.
ComboBox.scroll	Diffusion lorsque la liste déroulante est manipulée.

Hérite de tous les événements de la Classe UIObject, et de la Classe UIComponent.

ComboBox.addItem()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
OccurrenceDeListeDéroulante.addItem(étiquette[, données])
```

Usage 2 :

```
OccurrenceDeListeDéroulante.addItem({label:étiquette[, data:données]})
```

Usage 3 :

```
OccurrenceDeListeDéroulante.addItem(obj);
```

Paramètres

étiquette Chaîne indiquant l'étiquette du nouvel élément.

données Données de l'élément ; il peut s'agir de n'importe quel type de données. Ce paramètre est facultatif.

obj Objet possédant une propriété label et une propriété data facultative.

Renvoie

L'index auquel l'élément est ajouté.

Description

Méthode : ajoute un nouvel élément à la fin de la liste.

Exemple

Le code suivant ajoute un élément à l'occurrence `maListeDéroulante` :

```
maListeDéroulante.addItem("il s'agit d'un élément");
```

ComboBox.addItemAt()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
OccurrenceDeListeDéroulante.addItemAt(index, étiquette [, données])
```

Paramètres

index Nombre 0 ou supérieur indiquant la position à laquelle insérer l'élément (l'index du nouvel élément).

étiquette Chaîne indiquant l'étiquette du nouvel élément.

données Données de l'élément ; il peut s'agir de n'importe quel type de données. Ce paramètre est facultatif.

Renvoie

L'index auquel l'élément est ajouté.

Description

Méthode : ajoute un nouvel élément à la fin de la liste, à l'emplacement d'index spécifié par le paramètre *index*. Les index supérieurs à `ComboBox.length` sont ignorés.

Exemple

Le code suivant insère un élément à l'index 3, qui correspond à la quatrième place dans la liste déroulante (0 étant la première position) :

```
monChamp.addItemAt(3, "c'est le quatrième élément");
```

ComboBox.change

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

```
listenerObject = new Object();
objetDécoute.change = function(objetEvnt){
  // votre code ici
}
OccurrenceDeListeDéroulante.addEventListener("change", objetDécoute)
```

Description

Evénement : diffuse à tous les écouteurs enregistrés lorsque la valeur de la liste déroulante change suite à l'interaction d'un utilisateur.

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

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`OccurrenceDeListeDéroulante`*) distribue un événement (ici, `change`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `addEventListener()` (consultez `UIEventDispatcher.addEventListener()`) sur l'occurrence de composant qui diffuse l'événement afin d'enregistrer l'écouteur dans cette occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

L'exemple suivant envoie le nom de l'occurrence du composant ayant généré l'événement `change` au panneau de sortie :

```
form.change = function(objEvnt){
  trace("Valeur passée à " + objEvnt.target.value);
}
maListeDéroulante.addEventListener("change", form);
```

Consultez également

`UIEventDispatcher.addEventListener()`

ComboBox.close()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`maListeDéroulante``.close()`

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : ferme la liste déroulante.

Exemple

L'exemple suivant ferme la liste déroulante du composant ComboBox `monChamp` :

```
monChamp.close();
```

Consultez également

`ComboBox.open()`

ComboBox.close

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(close){
  // votre code ici
}
```

Usage 2 :

```
listenerObject = new Object();
objetDécoute.close = function(objetEvnt){
  // votre code ici
}
OccurrenceDeListeDéroulante.addEventListener("close", objetDécoute)
```

Description

Evénement : diffuse à tous les écouteurs enregistrés lorsque la liste déroulante commence à se rétracter.

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

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`OccurrenceDeListeDéroulante`*) distribue un événement (ici, `close`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

L'exemple suivant envoie un message au panneau de sortie lorsque la liste déroulante commence à se fermer :

```
form.close = function(){
  trace("La liste déroulante est fermée");
}
maListeDéroulante.addEventListener("close", form);
```

Consultez également

`UIEventDispatcher.addEventListener()`

ComboBox.dataProvider

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeListeDéroulante.dataProvider

Description

Propriété : modèle de données pour les éléments affichés dans une liste. La valeur de cette propriété peut être un tableau ou n'importe quel objet mettant en œuvre l'interface DataProvider. La valeur par défaut est []. Il s'agit d'une propriété du composant List, mais à laquelle il est possible d'accéder directement à partir d'une occurrence du composant ComboBox.

Le composant List et les autres composants de données ajoutent des méthodes au prototype de l'objet tableau pour être conformes à l'interface DataProvider (consultez DataProvider.as pour plus d'informations). Tout tableau qui existe parallèlement en tant que liste a donc automatiquement toutes les méthodes (`addItem()`, `getItemAt()`, etc.) nécessaires pour être le modèle d'une liste et peut être utilisé pour diffuser les changements de modèle vers plusieurs composants.

Si le tableau contient des objets, on accède aux propriétés `labelField` ou `labelFunction` pour déterminer quelles parties de l'élément doivent être affichées. La valeur par défaut étant `"label"`, si un champ de ce type existe, il est choisi pour être affiché ; dans le cas contraire, une liste de tous les champs séparés par une virgule est affichée.

Remarque : Si le tableau contient des chaînes et aucun objet à tous les emplacements d'index, la liste ne peut pas trier les éléments et conserver l'état de sélection. Le tri entraîne la perte de la sélection.

Toutes les occurrences mettant en œuvre l'interface DataProvider peuvent être choisies comme fournisseurs de données pour une liste. Cela inclut Flash Remoting RecordSets, Firefly DataSets, etc.

Exemple

Cet exemple utilise un tableau de chaînes utilisé pour remplir la liste déroulante :

```
comboBox.dataProvider = ["Expédition terrestre","Surlendemain, par
  avion","Lendemain, par avion"];
```

Cet exemple crée un tableau fournisseur de données et lui affecte la propriété dataProvider, comme suit :

```
monFD = new Array();
list.dataProvider = monFD;

pour (var i=0; i<accounts.length; i++) {
  // ces modifications apportées à DataProvider seront diffusées dans la liste
  monFD.addItem({ label: accounts[i].name,
                  data:  accounts[i].accountID });
}
```

ComboBox.dropdown

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
maListeDéroulante.dropdown
```

Description

Propriété (lecture seule) : renvoie une référence au composant List contenu dans ComboBox. Le sous-composant List n'est instancié dans le composant ComboBox que lorsqu'il doit être affiché. En revanche, dès que vous accédez à la propriété `dropdown`, la liste est créée.

Consultez également

`ComboBox.dropdownWidth`

ComboBox.dropdownWidth

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
maListeDéroulante.change
```

Description

Propriété : limite de la largeur en pixels pour la liste déroulante. La valeur par défaut est la largeur du composant ComboBox (l'occurrence TextInput plus l'occurrence SimpleButton).

Exemple

Le code suivant définit `dropdownWidth` à 150 pixels :

```
maListeDéroulante.dropdownWidth = 150;
```

Consultez également

`ComboBox.dropdown`

ComboBox.editable

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
maListeDéroulante.editable
```

Description

Propriété : indique si la liste déroulante est modifiable (`true`) ou non (`false`). Un composant ComboBox modifiable peut comporter des valeurs saisies dans la zone de texte mais qui ne seront pas affichées dans la liste déroulante. Si un composant ComboBox n'est pas modifiable, seules les valeurs répertoriées dans la liste déroulante peuvent être entrées dans la zone de texte. La valeur par défaut est `false`.

La définition d'une liste déroulante modifiable efface ce que contenait le champ de texte de la liste déroulante. Elle définit également l'index (et l'élément) sélectionné sur undefined. Pour rendre une liste déroulante modifiable tout en conservant l'élément sélectionné, utilisez le code suivant :

```
var ix = maListeDéroulante.selectedIndex;
maListeDéroulante.editable = true; // efface le texte dans le champ de texte.
maListeDéroulante.selectedIndex = ix; // copie de nouveau l'étiquette dans le
  champ de texte.
```

Exemple

Le code suivant rend `maListeDéroulante` modifiable :

```
maListeDéroulante.editable = true;
```

ComboBox.enter

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(enter){
  // votre code ici
}
```

Usage 2 :

```
listenerObject = new Object();
objetDécoute.enter = function(objetEvnt){
  // votre code ici
}
OccurrenceDeListeDéroulante.addEventListener("enter", objetDécoute)
```

Description

Evénement : diffuse à tous les écouteurs enregistrés lorsque la touche Entrée est enfoncée dans la zone de texte. Cet événement est uniquement diffusé à partir des listes déroulantes modifiables. Il s'agit d'un événement TextInput diffusé à partir d'une liste déroulante. Pour plus d'informations, consultez TextInput.enter.

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

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`OccurrenceDeListeDéroulante`*) distribue un événement (ici, `enter`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

L'exemple suivant envoie un message au panneau de sortie lorsque la liste déroulante commence à se fermer :

```
form.enter = function(){
  trace("L'événement enter de la liste déroulante a été déclenché");
}
maListe.addEventListener("enter", form);
```

Consultez également

`UIEventDispatcher.addEventListener()`

ComboBox.getItemAt()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`OccurrenceDeListeDéroulante``.getItemAt(`*`index`*`)`

Paramètres

`index` Nombre supérieur ou égal à 0, et inférieur à `ComboBox.length`. Index de l'élément à récupérer.

Renvoie

Valeur ou objet d'élément indexé. La valeur n'est pas définie si l'index est en dehors des limites.

Description

Méthode : récupère l'élément à l'emplacement d'index spécifié.

Exemple

Le code suivant affiche l'élément à l'emplacement d'index 4 :

```
trace(monChamp.getItemAt(4).label);
```

ComboBox.itemRollOut

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(itemRollOut){
  // votre code ici
}
```

Usage 2 :

```
listenerObject = new Object();
objetDécoute.itemRollOut = function(ObjetEvnt){
  // votre code ici
}
OccurrenceDeListeDéroulante.addEventListener("itemRollOut", objetDécoute)
```

Objet événement

Outre les propriétés standard de l'objet événement, l'événement `itemRollOut` possède une propriété supplémentaire : `index`. L'`index` correspond au numéro de l'élément à la sortie du survol.

Description

Evénement : diffuse à tous les écouteurs enregistrés lorsque le pointeur cesse de survoler les éléments de la liste déroulante. Il s'agit d'un événement List diffusé à partir d'une liste déroulante. Pour plus d'informations, consultez `List.itemRollOut`.

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

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*OccurrenceDeListeDéroulante*) distribue un événement (ici, `itemRollOut`) qui est géré par une fonction associée à un objet d'écoute (*objetDécoute*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*objetEvnt*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Pour finir, vous appelez la méthode `addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Exemple

L'exemple suivant envoie un message au panneau de sortie indiquant le numéro d'index à la sortie du survol :

```
form.itemRollOut = function (objEvnt) {
  trace("Item #" + objEvnt.index + " à la sortie du survol.");
}
maListe.addEventListener("itemRollOut", form);
```

Consultez également

`ComboBox.itemRollOver`, `UIEventDispatcher.addEventListener()`

ComboBox.itemRollOver

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(itemRollOver){
  // votre code ici
}
```

Usage 2 :

```
listenerObject = new Object();
objetDécoute.itemRollOver = function(ObjetEvnt){
  // votre code ici
}
OccurrenceDeListeDéroulante.addEventListener("itemRollOver", objetDécoute)
```

Objet événement

Outre les propriétés standard de l'objet événement, l'événement `itemRollOver` possède une propriété supplémentaire : `index`. L'`index` est le numéro de l'élément survolé par le pointeur.

Description

Evénement : diffuse à tous les écouteurs enregistrés lorsque les éléments de la liste déroulante sont survolés. Il s'agit d'un événement List diffusé à partir d'une liste déroulante. Pour plus d'informations, consultez List.itemRollOver.

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

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`OccurrenceDeListeDéroulante`*) distribue un événement (ici, `itemRollOver`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Pour finir, vous appelez la méthode `addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Exemple

L'exemple suivant envoie un message au panneau de sortie indiquant le numéro d'index survolé par le pointeur :

```
form.itemRollOver = function (objEvnt) {
  trace("Item #" + objEvnt.index + " survolé.");
}
maListe.addEventListener("itemRollOver", form);
```

Consultez également

`ComboBox.itemRollOut`, `UIEventDispatcher.addEventListener()`

ComboBox.labelField

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`maListeDéroulante``.labelField`

Description

Propriété : le nom du champ dans les objets du tableau dataProvider à utiliser comme champ d'étiquette. Il s'agit de la propriété du composant List disponible à partir d'une occurrence de composant ComboBox. Pour plus d'informations, consultez `List.labelField`.

La valeur par défaut est undefined.

Exemple

L'exemple suivant définit la propriété `dataProvider` sur un tableau de chaînes et configure la propriété `labelField` afin d'indiquer que le champ `nom` doit être utilisé comme étiquette pour la liste déroulante :

```
maListeDéroulante.dataProvider  = [
  {name:"Gabriel", gender:"masculin"},
  {name:"Susanne", gender:"féminin"} ];

maListeDéroulante.labelField = "nom";
```

Consultez également

`List.labelFunction`

ComboBox.labelFunction

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`maListeDéroulante``.labelFunction`

Description

Propriété : fonction qui calcule l'étiquette d'un objet dataProvider. Vous devez définir la fonction. La valeur par défaut est undefined.

Exemple

L'exemple suivant crée un fournisseur de données, puis définit une fonction afin de spécifier l'objet à utiliser comme étiquette dans la liste déroulante :

```
maListeDéroulante.dataProvider  = [
  {firstName:"Nicolas", lastName:"Petit", age:"très jeune"},
  {firstName:"Gabriel", lastName:"Grosjean", age:"jeune"},
  {firstName:"Christian", lastName:"Valoin", age:"âgé"},
  {firstName:"Grégoire", lastName:"Martin", age:"très âgé"} ];

maListeDéroulante.labelFunction = function(itemObj){
  return (itemObj.lastName + ", " + itemObj.firstName);
}
```

Consultez également

`List.labelField`

ComboBox.length

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

maListeDéroulante.length

Description

Propriété (lecture seule) : longueur de la liste déroulante. Il s'agit d'une propriété du composant List disponible à partir d'une occurrence de composant ComboBox. Pour plus d'informations, consultez `List.length`. La valeur par défaut est 0.

Exemple

L'exemple suivant stocke la valeur de `length` dans une variable :

```
dropdownItemCount = monChamp.length;
```

ComboBox.open()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

maListeDéroulante.open()

Paramètres

Aucun.

Renvoie

Rien.

Description

Propriété : ouvre la liste déroulante.

Exemple

Le code suivant ouvre la liste déroulante pour l'occurrence `combo1` :

```
combo1.open();
```

Consultez également

`ComboBox.close()`

ComboBox.open

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(open){
  // votre code ici
}
```

Usage 2 :

```
listenerObject = new Object();
objetDécoute.open = function(ObjetEvnt){
  // votre code ici
}
OccurrenceDeListeDéroulante.addEventListener("open", objetDécoute)
```

Description

Evénement : diffuse à tous les écouteurs enregistrés lorsque la liste déroulante commence à apparaître.

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

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`OccurrenceDeListeDéroulante`*) distribue un événement (ici, `open`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Pour finir, vous appelez la méthode `addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Exemple

L'exemple suivant envoie un message au panneau de sortie indiquant le numéro d'index qui vient d'être survolé par le pointeur :

```
form.open = function () {
```

```
  trace("La liste déroulante s'est ouverte avec le texte " + monChamp.text);
}
monChamp.addEventListener("open", form);
```

Consultez également

ComboBox.close, UIEventDispatcher.addEventListener()

ComboBox.removeAll()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeListeDéroulante.removeAll()

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : supprime tous les éléments de la liste. Il s'agit d'une propriété du composant List disponible à partir d'une occurrence de composant ComboBox.

Exemple

Le code suivant supprime tous les éléments de la liste :

```
maListeDéroulante.removeAll();
```

Consultez également

ComboBox.removeItemAt(), ComboBox.replaceItemAt()

ComboBox.removeItemAt()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceListe.removeItemAt(*index*)

Paramètres

index Nombre indiquant la position de l'élément à supprimer. Cette valeur est basée sur zéro.

Renvoie

Un objet : l'élément supprimé (undefined si aucun élément n'existe).

Description

Méthode : supprime un élément à l'emplacement d'index indiqué. Un index disparaît parmi les index de la liste situés après l'index indiqué par le paramètre *index*. Il s'agit d'une propriété du composant List disponible à partir d'une occurrence de composant ComboBox.

Exemple

Le code suivant supprime l'élément à l'emplacement d'index 3 :

```
maListeDéroulante.removeItemAt(3);
```

Consultez également

ComboBox.removeAll(), ComboBox.replaceItemAt()

ComboBox.replaceItemAt()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
OccurrenceDeListeDéroulante.replaceItemAt(index, étiquette [,data])
```

Paramètres

index Nombre 0 ou supérieur indiquant la position à laquelle insérer l'élément (l'index du nouvel élément).

étiquette Chaîne indiquant l'étiquette du nouvel élément.

données Données de l'élément. Ce paramètre est facultatif.

Renvoie

Rien.

Description

Méthode : remplace le contenu de l'élément à l'emplacement d'index spécifié par le paramètre *index*. Il s'agit d'une méthode du composant List disponible à partir d'une occurrence de composant ComboBox.

Exemple

L'exemple suivant modifie la troisième place d'index :

```
maListeDéroulante.replaceItemAt(3, "new label");
```

Consultez également

ComboBox.removeAll(), ComboBox.removeItemAt()

ComboBox.rowCount

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
maListeDéroulante.rowCount
```

Description

Propriété : nombre maximal de rangées visibles dans la liste déroulante. La valeur par défaut est 5.

Si le nombre d'éléments de la liste déroulante est supérieur ou égal à la propriété `rowCount`, la liste est redimensionnée et une barre de défilement est affichée si nécessaire. Si la liste déroulante contient un nombre d'éléments inférieur par rapport à la propriété `rowCount`, elle est redimensionnée en fonction du nombre d'éléments contenus.

Ce comportement est différent du composant List, qui affiche toujours le nombre de rangées spécifié par sa propriété `rowCount`, même en cas d'espaces vides.

Si la valeur est négative ou décimale, le comportement n'est pas défini.

Exemple

L'exemple suivant spécifie que la liste déroulante doit contenir un maximum de 20 rangées visibles :

```
maListeDéroulante.rowCount = 20;
```

ComboBox.scroll

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(scroll){
  // votre code ici
}
```

Usage 2 :

```
listenerObject = new Object();
objetDécoute.scroll = function(ObjetEvnt){
  // votre code ici
}
OccurrenceDeListeDéroulante.addEventListener("scroll", objetDécoute)
```

Objet événement

Outre les propriétés standard de l'objet événement, l'événement scroll possède une propriété supplémentaire : `direction`. Il s'agit d'une chaîne ayant deux valeurs possibles : `"horizontal"` ou `"vertical"`. Pour un événement `scroll` ComboBox, la valeur est toujours `"vertical"`.

Description

Evénement : diffuse à tous les écouteurs enregistrés lorsque la liste déroulante défile. Il s'agit d'un événement de composant List disponible pour le composant ComboBox.

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

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`OccurrenceDeListeDéroulante`*) distribue un événement (ici, `scroll`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Pour finir, vous appelez la méthode `addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Exemple

L'exemple suivant envoie un message au panneau de sortie indiquant le numéro d'index atteint :

```
form.scroll = function(objEvnt){
  trace("La liste affiche l'élément " + objEvnt.target.vPosition);
}
maListeDéroulante.addEventListener("scroll", form);
```

Consultez également

`UIEventDispatcher.addEventListener()`

ComboBox.selectedIndex

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

maListeDéroulante.selectedIndex

Description

Propriété : l'index (numéro) de l'élément sélectionné dans la liste déroulante. La valeur par défaut est 0. L'affectation de cette propriété provoque la suppression de l'élément sélectionné, entraîne la sélection de l'élément indiqué et affiche l'étiquette de l'élément sélectionné dans la zone de texte du composant ComboBox.

L'affectation d'un `selectedIndex` en dehors des limites est ignorée. La saisie de texte dans le champ de texte d'une liste déroulante modifiable définit `selectedIndex` sur undefined.

Exemple

Le code suivant sélectionne le dernier élément de la liste :

```
maListeDéroulante.selectedIndex = maListeDéroulante.length-1;
```

Consultez également

ComboBox.selectedItem

ComboBox.selectedItem

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

maListeDéroulante.selectedItem

Description

Propriété : valeur de l'élément sélectionné dans la liste déroulante.

Si la liste déroulante est modifiable, `selectedItem` renvoie la valeur « undefined » si vous saisissez du texte dans la zone de texte. Il n'y a de valeur que si vous sélectionnez un élément dans la liste déroulante ou si la valeur est définie via ActionScript. Si la liste déroulante est statique, la valeur de `selectedItem` est toujours valide.

Exemple

L'exemple suivant affiche `selectedItem` si le fournisseur de données contient des types primitifs :

```
var item = maListeDéroulante.selectedItem;
trace("Vous avez sélectionné l'élément " + item);
```

L'exemple suivant affiche `selectedItem` si le fournisseur de données contient des objets ayant des propriétés `label` et `data` :

```
var obj = maListeDéroulante.selectedItem;
trace("Vous avez sélectionné la couleur intitulée : " + obj.label);
trace("La valeur hexadécimale de cette couleur est : " + obj.data);
```

Consultez également

ComboBox.dataProvider, ComboBox.selectedIndex

ComboBox.text

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

maListeDéroulante.text

Description

Propriété : le texte de la zone de texte. Vous pouvez obtenir et définir cette valeur pour les listes déroulantes modifiables. Pour les listes déroulantes statiques, la valeur est en lecture seule.

Exemple

L'exemple suivant définit la valeur `text` actuelle d'une liste déroulante modifiable :

```
maListeDéroulante.text = "Californie";
```

ComboBox.textField

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

maListeDéroulante.textField

Description

Propriété (lecture seule) : référence au composant TextInput contenu dans ComboBox.

Cette propriété vous permet d'accéder au composant TextInput situé en dessous pour le manipuler. Par exemple, vous pouvez changer la sélection de la zone de texte ou restreindre les caractères pouvant y être saisis.

Exemple

Le code suivant restreint la zone de texte de `maListeDéroulante` à la seule saisie des chiffres :

```
maListeDéroulante.textField.restrict = "0-9";
```

ComboBox.value

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

maListeDéroulante.value

Description

Propriété (lecture seule) : si la liste déroulante est modifiable, `value` renvoie la valeur de la zone de texte. Si la liste déroulante est statique, `value` renvoie la valeur de la liste. La valeur de la liste est le champ `data` ou, si le champ `data` n'existe pas, le champ `label`.

Exemple

L'exemple suivant place les données dans la liste déroulante en définissant la propriété `dataProvider`. Il affiche ensuite la `valeur` dans le panneau de sortie. Enfin, il sélectionne `"Californie"` et l'affiche dans la zone de texte, comme suit :

```
cb.dataProvider = [
  {label:"Alaska", data:"AZ"},
  {label:"Californie", data:"CA"},
  {label:"Washington", data:"WA"}];

cb.editable = true;
cb.selectedIndex = 1;
trace('La valeur modifiable est "Californie": '+ cb.value);

cb.editable = false;
cb.selectedIndex = 1;
trace('La valeur non modifiable est "CA": '+ cb.value);
```

Paquet DataBinding

Pour des informations actualisées sur cette fonction, cliquez sur le bouton Mettre à jour, dans la partie supérieure de l'onglet Aide.

Composant DataGrid

Pour obtenir les informations les plus récentes sur cette fonction, cliquez sur le bouton Mettre à jour dans la partie supérieure de l'onglet Aide.

Composant DataHolder

Pour des informations actualisées sur cette fonction, cliquez sur le bouton Mettre à jour, dans la partie supérieure de l'onglet Aide.

Composant DataProvider

Pour des informations actualisées sur cette fonction, cliquez sur le bouton Mettre à jour, dans la partie supérieure de l'onglet Aide.

Composant DataSet

Pour des informations actualisées sur cette fonction, cliquez sur le bouton Mettre à jour, dans la partie supérieure de l'onglet Aide.

Composant DateChooser

Pour obtenir les informations les plus récentes sur cette fonction, cliquez sur le bouton Mettre à jour dans la partie supérieure de l'onglet Aide.

Composant DateField

Pour des informations actualisées sur cette fonction, cliquez sur le bouton Mettre à jour, dans la partie supérieure de l'onglet Aide.

Classe DepthManager

Espace de nom de classe ActionScript mx.managers.DepthManager

La classe DepthManager, qui complète la classe ActionScript MovieClip, vous permet de gérer les affectations de profondeur relatives de tous les composants ou clips, y compris `_root`. Elle vous autorise également à gérer les profondeurs réservées dans un clip spécial, et ceci à la profondeur la plus élevée sur `_root` pour les services de niveau système, tels que le curseur ou les info-bulles.

L'API de tri de profondeur relative se compose des méthodes suivantes :

- `DepthManager.createChildAtDepth()`
- `DepthManager.createClassChildAtDepth()`
- `DepthManager.setDepthAbove()`
- `DepthManager.setDepthBelow()`
- `DepthManager.setDepthTo()`

Les méthodes suivantes composent les API d'espace de profondeur réservée :

- `DepthManager.createClassObjectAtDepth()`
- `DepthManager.createObjectAtDepth()`

Méthodes de la classe DepthManager

Méthode	Description
`DepthManager.createChildAtDepth()`	Crée un enfant du symbole indiqué à la profondeur spécifiée.
`DepthManager.createClassChildAtDepth()`	Crée un objet de la classe indiquée à la profondeur spécifiée.
`DepthManager.createClassObjectAtDepth()`	Crée une occurrence de la classe indiquée à la profondeur spécifiée dans le clip spécial de profondeur la plus élevée.
`DepthManager.createObjectAtDepth()`	Crée un objet à une profondeur spécifiée dans le clip spécial de profondeur la plus élevée.
`DepthManager.setDepthAbove()`	Définit la profondeur au-dessus de l'occurrence spécifiée.
`DepthManager.setDepthBelow()`	Définit la profondeur sous l'occurrence spécifiée.
`DepthManager.setDepthTo()`	Définit la profondeur de l'occurrence spécifiée dans le clip de profondeur la plus élevée.

DepthManager.createChildAtDepth()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceClip.createChildAtDepth(*nomLiaison*, *drapeauProfondeur*[, *objInit*])

Paramètres

nomLiaison Identificateur de liaison. Ce paramètre est une chaîne.

drapeauProfondeur L'une des valeurs suivantes : DepthManager.kTop, DepthManager.kBottom, DepthManager.kTopmost, DepthManager.kNotopmost. Tous les drapeaux de profondeur sont des propriétés statiques de la classe DepthManager. Vous devez faire référence au paquet DepthManager (mx.managers.DepthManager.kTopmost, par exemple) ou utiliser l'instruction import pour importer le paquet DepthManager.

objInit Objet d'initialisation. Ce paramètre est facultatif.

Renvoie

Référence à l'objet créé.

Description

Méthode : crée une occurrence enfant du symbole spécifié par le paramètre *nomLiaison* à la profondeur spécifiée par le paramètre *drapeauProfondeur*.

Exemple

L'exemple suivant crée une occurrence grandeAiguille du clip SymboleMinute et la place sur l'horloge :

```
import mx.managers.DepthManager;
grandeAiguille = clock.createChildAtDepth("SymboleMinute", DepthManager.kTop);
```

DepthManager.createClassChildAtDepth()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004 et Flash MX Professionnel 2004

Usage

occurrenceClip.createClassChildAtDepth(*nomClasse*, *drapeauProfondeur*, *objInit*)

Paramètres

nomClasse Nom de classe.

drapeauProfondeur L'une des valeurs suivantes : `DepthManager.kTop`, `DepthManager.kBottom`, `DepthManager.kTopmost`, `DepthManager.kNotopmost`. Tous les drapeaux de profondeur sont des propriétés statiques de la classe DepthManager. Vous devez faire référence au paquet DepthManager (`mx.managers.DepthManager.kTopmost`, par exemple) ou utiliser l'instruction `import` pour importer le paquet DepthManager.

objInit Objet d'initialisation. Ce paramètre est facultatif.

Renvoie

Référence à l'enfant créé.

Description

Méthode : crée un enfant de la classe spécifiée par le paramètre *nomClasse* à la profondeur spécifiée par le paramètre *drapeauProfondeur*.

Exemple

Le code suivant dessine un rectangle de focus par-dessus tous les objets NoTopmost :

```
import mx.managers.DepthManager
this.ring = createClassChildAtDepth(mx.skins.RectBorder, DepthManager.kTop);
```

Le code suivant crée une occurrence de la classe Button et lui transmet une valeur pour sa propriété `label` en tant que paramètre *objInit* :

```
import mx.managers.DepthManager
button1 = createClassChildAtDepth(mx.controls.Button, DepthManager.kTop,
  {label: "Bouton supérieur"});
```

DepthManager.createClassObjectAtDepth()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004 et Flash MX Professionnel 2004

Usage

```
DepthManager.createClassObjectAtDepth(nomClasse, espaceProfondeur[, objInit])
```

Paramètres

nomClasse Nom de classe.

espaceProfondeur L'une des valeurs suivantes : `DepthManager.kCursor`, `DepthManager.kTooltip`. Tous les drapeaux de profondeur sont des propriétés statiques de la classe DepthManager. Vous devez faire référence au paquet DepthManager (`mx.managers.DepthManager.kCursor`, par exemple) ou utiliser l'instruction `import` pour importer le paquet DepthManager.

objInit Objet d'initialisation. Ce paramètre est facultatif.

Renvoie

Référence à l'objet créé.

Description

Méthode : crée un objet de la classe spécifiée par le paramètre *nomClasse* à la profondeur spécifiée par le paramètre *espaceProfondeur*. Cette méthode est utilisée pour accéder aux espaces de profondeur réservés dans le clip spécial de profondeur la plus élevée.

Exemple

L'exemple suivant crée un objet à partir de la classe Button :

```
import mx.managers.DepthManager
monBoutonCurseur = createClassObjectAtDepth(mx.controls.Button,
  DepthManager.kCursor, {label: "Curseur"});
```

DepthManager.createObjectAtDepth()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004 et Flash MX Professionnel 2004

Usage

```
DepthManager.createObjectAtDepth(nomLiaison, espaceProfondeur[, objInit])
```

Paramètres

nomLiaison Identificateur de liaison.

espaceProfondeur L'une des valeurs suivantes : `DepthManager.kCursor`, `DepthManager.kTooltip`. Tous les drapeaux de profondeur sont des propriétés statiques de la classe DepthManager. Vous devez faire référence au paquet DepthManager (`mx.managers.DepthManager.kCursor`, par exemple) ou utiliser l'instruction `import` pour importer le paquet DepthManager.

objInit Objet d'initialisation.

Renvoie

Référence à l'objet créé.

Description

Méthode : crée un objet à la profondeur spécifiée. Cette méthode est utilisée pour accéder aux espaces de profondeur réservés dans le clip spécial de profondeur la plus élevée.

Exemple

L'exemple suivant crée une occurrence du symbole SymboleInfoBulle et la place à la profondeur réservée aux info-bulles :

```
import mx.managers.DepthManager
monInfobulleCurseur = createObjectAtDepth("SymboleInfoBulle",
  DepthManager.kTooltip);
```

DepthManager.setDepthAbove()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004 et Flash MX Professionnel 2004

Usage

```
occurrenceClip.setDepthAbove(instance)
```

Paramètres

instance Nom d'occurrence.

Renvoie

Rien.

Description

Méthode : définit la profondeur d'une occurrence de clip ou de composant au-dessus de la profondeur de l'occurrence spécifiée par le paramètre *instance*.

DepthManager.setDepthBelow()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004 et Flash MX Professionnel 2004

Usage

```
occurrenceClip.setDepthBelow(instance)
```

Paramètres

instance Nom d'occurrence.

Renvoie

Rien.

Description

Méthode : définit la profondeur d'une occurrence de clip ou de composant en dessous de la profondeur de l'occurrence spécifiée par le paramètre *instance*.

Exemple

Le code suivant définit la profondeur de l'occurrence `textInput` au-dessous de la profondeur de `button` :

```
textInput.setDepthBelow(button);
```

DepthManager.setDepthTo()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004 et Flash MX Professionnel 2004

Usage

```
occurrenceClip.setDepthTo(profondeur)
```

Paramètres

profondeur Niveau de profondeur

Renvoie

Rien.

Description

Méthode : définit la profondeur de *occurrenceClip* sur la valeur spécifiée par *profondeur*. Cette méthode place une occurrence à une autre profondeur afin de faire de la place pour un autre objet.

Exemple

L'exemple suivant fixe la profondeur de l'occurrence `mc1` à 10 :

```
mc1.setDepthTo(10);
```

Pour plus d'informations sur la profondeur et l'ordre d'empilement, consultez « Définition de la prochaine profondeur maximale disponible » dans le Dictionnaire ActionScript de l'aide.

Classe FocusManager

Vous pouvez utiliser FocusManager pour spécifier l'ordre dans lequel les composants reçoivent le focus lorsqu'un utilisateur appuie sur la touche de tabulation pour parcourir une application. Vous pouvez utiliser l'API de FocusManager pour définir un bouton dans votre document qui recevra les entrées de clavier lorsque l'utilisateur appuiera sur Entrée (Windows) ou Retour (Macintosh). Par exemple, lorsqu'un utilisateur remplit un formulaire, il doit pouvoir appuyer sur la touche de tabulation pour passer d'un champ à l'autre et appuyer sur Entrée (Windows) ou Retour (Macintosh) pour envoyer le formulaire.

Tous les composants supportent FocusManager ; vous n'avez pas besoin de rédiger du code pour l'invoquer. FocusManager établit également une interaction avec le Gestionnaire système, qui active et désactive les occurrences FocusManager lorsque les fenêtres contextuelles sont activées ou désactivées. Chaque fenêtre modale a une occurrence de FocusManager pour que ses composants définissent eux-mêmes la tabulation et empêchent ainsi l'utilisateur d'aller dans les composants des autres fenêtres avec la touche de tabulation.

`RadioButton.groupName`FocusManager reconnaît les groupes de boutons radio (ceux dont la propriété est définie) et règle le focus sur l'occurrence du groupe dont la propriété `selected` correspond à `true`. Lorsque la touche de tabulation est enfoncée, le Gestionnaire de focus vérifie si l'objet suivant a le même `groupName` que l'objet en cours. Si tel est le cas, il déplace automatiquement le focus sur l'objet suivant ayant un `groupName` différent. Les autres jeux de composants qui supportent une propriété `groupName` peuvent également utiliser cette fonction.

FocusManager traite les changements de focus provoqués par les clics de la souris. Si l'utilisateur clique sur un composant, celui-ci reçoit le focus.

Le Gestionnaire de focus n'affecte pas automatiquement de focus à un composant d'une application. Dans la fenêtre principale ou toute fenêtre contextuelle, le focus n'est pas placé par défaut sur un composant. Pour cela, vous devez appeler `focusManager.setFocus` sur un composant.

Utilisation de FocusManager

Pour créer une navigation de focus dans une application, définissez la propriété `tabIndex` sur tous les objets (y compris les boutons) qui doivent recevoir le focus. Lorsqu'un utilisateur appuie sur la touche de tabulation, FocusManager recherche un objet activé ayant une propriété `tabIndex` supérieure à la valeur actuelle de `tabIndex`. Une fois que FocusManager a trouvé la propriété `tabIndex` la plus élevée, il retombe à zéro. Dans l'exemple suivant, l'objet `commentaire` (probablement un composant TextArea) reçoit le focus le premier, puis c'est au tour de l'objet `boutonOK` d'en recevoir :

```
comment.tabIndex = 1;
okButton.tabIndex = 2;
```

Pour créer un bouton qui reçoive le focus lorsqu'un utilisateur appuie sur Entrée (Windows) ou sur Retour (Macintosh), définissez la propriété `FocusManager.defaultPushButton` sur le nom d'occurrence du bouton désiré, comme dans l'exemple suivant :

```
focusManager.defaultPushButton = okButton;
```

Remarque : FocusManager réagit au moment où les objets sont placés sur la scène (l'ordre de profondeur des objets), et non à leur position relative sur la scène. C'est ce qui le distingue de la manière dont Flash Player traite les tabulations.

Paramètres de FocusManager

Il n'existe pas de paramètres de création pour FocusManager. Vous devez utiliser les méthodes ActionScript et les propriétés de la classe FocusManager dans le panneau Actions. Pour plus d'informations, consultez Classe FocusManager.

Création d'une application avec FocusManager

La procédure suivante crée un programme de focus dans une application Flash.

1. Faites glisser le composant TextInput du panneau Composants jusqu'à la scène.
2. Dans l'inspecteur des propriétés, affectez-lui le nom d'occurrence **commentaire**.
3. Faites glisser le composant Button du panneau Composants jusqu'à la scène.
4. Dans l'inspecteur des propriétés, affectez-lui le nom d'occurrence **boutonOK** et définissez le paramètre d'étiquette sur **OK**.

5 Dans l'image 1 du panneau Actions, saisissez le code suivant :

```
comment.tabIndex = 1;
okButton.tabIndex = 2;
focusManager.setFocus(comment);
focusManager.defaultPushButton = okButton;
lo = new Object();
lo.click = function(){
  trace("bouton cliqué");
}
boutonOK.addEventListener("click", lo);
```

Ce code définit l'ordre des tabulations et spécifie un bouton par défaut qui recevra un événement `click` lorsqu'un utilisateur appuiera sur Entrée (Windows) ou Retour (Macintosh).

Personnalisation de FocusManager

Vous pouvez changer la couleur du cercle du focus dans le thème Halo en modifiant la valeur du style `themeColor`.

FocusManager utilise une enveloppe FocusRect pour le tracé du focus. Cette enveloppe peut être remplacée ou modifiée et les sous-classes peuvent se substituer à `UIComponent.drawFocus` pour le tracé des indicateurs de focus personnalisés.

Classe FocusManager

Héritage UIObject > UIComponent > FocusManager

Espace de nom de classe ActionScript mx.managers.FocusManager

Méthodes de la classe FocusManager

Méthode	Description
`FocusManager.getFocus()`	Renvoie une référence à l'objet ayant le focus.
`FocusManager.sendDefaultPushButtonEvent()`	Envoie un événement `click` aux objets d'écoute enregistrés sur le bouton-poussoir par défaut.
`FocusManager.setFocus()`	Définit le focus sur l'objet spécifié.

Propriétés de la classe FocusManager

Méthode	Description
`FocusManager.defaultPushButton`	Objet qui reçoit un événement `click` lorsqu'un utilisateur appuie sur la touche Retour ou Entrée.
`FocusManager.defaultPushButtonEnabled`	Indique si le traitement clavier du bouton-poussoir par défaut est activé (`true`) ou désactivé (`false`). La valeur par défaut est `true`.
`FocusManager.enabled`	Indique si le traitement tabulation est activé (`true`) ou désactivé (`false`). La valeur par défaut est `true`.
`FocusManager.nextTabIndex`	Valeur suivante de la propriété `tabIndex`.

FocusManager.defaultPushButton

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004 et Flash MX Professionnel 2004.

Usage

```
focusManager.defaultPushButton
```

Description

Propriété : spécifie le bouton-poussoir par défaut pour une application. Lorsque l'utilisateur appuie sur la touche Entrée (Windows) ou Retour (Macintosh), les écouteurs du bouton-poussoir par défaut reçoivent un événement `click`. La valeur par défaut n'est pas définie et les données de cette propriété sont de type objet.

FocusManager utilise la déclaration de style de mise en valeur de la classe SimpleButton pour identifier visuellement le bouton-poussoir par défaut.

La valeur de la propriété `defaultPushButton` est toujours le bouton qui a le focus. La définition de la propriété `defaultPushButton` ne donne pas le focus initial au bouton-poussoir par défaut. Si une application comprend plusieurs boutons, celui qui a le focus reçoit l'événement `click` lorsque la touche Entrée ou Retour est enfoncée. Si le focus se trouve sur un autre composant lorsque la touche Entrée ou Retour est enfoncée, la valeur d'origine de la propriété `defaultPushButton` est rétablie.

Exemple

Le code suivant définit le bouton-poussoir par défaut sur l'occurrence `boutonOK` :

```
FocusManager.defaultPushButton = boutonOK;
```

Consultez également

`FocusManager.defaultPushButtonEnabled`,
`FocusManager.sendDefaultPushButtonEvent()`

FocusManager.defaultPushButtonEnabled

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
focusManager.defaultPushButtonEnabled
```

Description

Propriété : valeur booléenne qui détermine si le traitement clavier du bouton-poussoir par défaut est activé (`true`) ou non (`false`). La définition de `defaultPushButtonEnabled` sur `false` permet à un composant de recevoir la touche Retour ou Entrée et de la traiter en interne. Vous devez réactiver le traitement du bouton-poussoir par défaut en suivant la méthode `onKillFocus()` du composant (consultez `MovieClip.onKillFocus` dans le Dictionnaire ActionScript de l'aide) ou l'événement `focusOut`. La valeur par défaut est `true`.

Exemple

Le code suivant désactive le traitement du bouton-poussoir par défaut :

```
focusManager.defaultPushButtonEnabled = false;
```

FocusManager.enabled

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
focusManager.enabled
```

Description

Propriété : valeur booléenne qui détermine si le traitement tabulation est activé (`true`) ou non (`false`) pour un groupe spécifique d'objets de focus. Une autre fenêtre contextuelle pourrait avoir son propre FocusManager, par exemple. La définition de `enabled` sur `false` permet à un composant de recevoir les touches de traitement tabulation et de les traiter en interne. Vous devez réactiver le traitement FocusManager en suivant la méthode `onKillFocus()` du composant (consultez `MovieClip.onKillFocus` dans le Dictionnaire ActionScript de l'aide) ou l'événement `focusOut`. La valeur par défaut est `true`.

Exemple

Le code suivant désactive la tabulation :

```
focusManager.enabled = false;
```

FocusManager.getFocus()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004 et Flash MX Professionnel 2004

Usage

```
focusManager.getFocus()
```

Paramètres

Aucun.

Renvoie

Référence à l'objet ayant le focus.

Description

Méthode : renvoie une référence à l'objet ayant le focus.

Exemple

Le code suivant définit le focus sur `monBoutonOK` si l'objet ayant le focus est `maSaisieDeTexte` :

```
si (focusManager.getFocus() == maSaisieDeTexte)
{
  focusManager.setFocus(monBoutonOK);
}
```

Consultez également

`FocusManager.setFocus()`

FocusManager.nextTabIndex

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
FocusManager.nextTabIndex
```

Description

Propriété : le numéro suivant d'index de tabulation disponible. Utilisez cette propriété pour définir dynamiquement la propriété `tabIndex` d'un objet.

Exemple

Le code suivant donne à l'occurrence `maCaseAcocher` la valeur `tabIndex` suivante la plus élevée :

```
maCaseAcocher.tabIndex = focusManager.nextTabIndex;
```

Consultez également

`UIComponent.tabIndex`

FocusManager.sendDefaultPushButtonEvent()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004 et Flash MX Professionnel 2004

Usage

```
focusManager.sendDefaultPushButtonEvent()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : envoie un événement `click` aux objets d'écoute enregistrés sur le bouton-poussoir par défaut. Utilisez cette méthode pour envoyer par programmation un événement `click`.

Exemple

Le code suivant déclenche l'événement `click` de bouton-poussoir par défaut et remplit les champs de nom d'utilisateur et de mot de passe lorsqu'un utilisateur sélectionne l'occurrence CheckBox `chb` (la case à cocher s'appellerait « Connexion automatique ») :

```
name_txt.tabIndex = 1;
password_txt.tabIndex = 2;
chb.tabIndex = 3;
submit_ib.tabIndex = 4;

focusManager.defaultPushButton = submit_ib;

chbObj = new Object();
chbObj.click = function(o){
  if (chb.selected == true){
    name_txt.text = "Lucie";
    password_txt.text = "secret";
    focusManager.sendDefaultPushButtonEvent();
} else {
    name_txt.text = "";
    password_txt.text = "";
  }
}
chb.addEventListener("click", chbObj);

submitObj = new Object();
submitObj.click = function(o){
  if (password_txt.text != "secret"){
    trace("erreur de soumission");
} else {
    trace("sendDefaultPushButtonEvent a abouti !");
  }
}
submit_ib.addEventListener("click", submitObj);
```

Consultez également

`FocusManager.defaultPushButton`, `FocusManager.sendDefaultPushButtonEvent()`

FocusManager.setFocus()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004 et Flash MX Professionnel 2004

Usage

```
focusManager.setFocus(object)
```

Paramètres

object Référence à l'objet qui reçoit le focus.

Renvoie

Rien.

Description

Méthode : définit le focus sur l'objet spécifié.

Exemple

Le code suivant définit le focus sur `monBoutonOK` :

```
focusManager.setFocus(monBoutonOK);
```

Consultez également

`FocusManager.getFocus()`

Classe Form

Pour des informations actualisées sur cette fonction, cliquez sur le bouton Mettre à jour, dans la partie supérieure de l'onglet Aide.

Composant Label

Un composant d'étiquette est une ligne unique de texte. Vous pouvez spécifier qu'une étiquette devra être formatée avec HTML. Vous pouvez également contrôler l'alignement et le dimensionnement des étiquettes. Les étiquettes n'ont pas de bordures, ne peuvent pas recevoir le focus et ne diffusent pas d'événements.

Un aperçu en direct de chaque occurrence d'étiquette reflète les modifications apportées à leurs paramètres dans le panneau de l'inspecteur des propriétés ou des composants au cours de la programmation. Le composant Label n'ayant pas de bordures, la définition de son paramètre de texte constitue le seul moyen de visualiser son aperçu en direct. Si le texte est trop long et que vous choisissez de définir le paramètre autoSize, celui-ci n'est pas pris en charge par l'aperçu en direct et le cadre de délimitation de l'étiquette n'est pas redimensionné. Vous devez cliquer à l'intérieur du cadre de délimitation pour sélectionner l'étiquette sur la scène.

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

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

Vous n'activez l'accessibilité d'un composant qu'une fois, quel que soit son nombre d'occurrences. Pour plus d'informations, consultez « Création de contenu accessible », dans le guide Utilisation de Flash de l'aide. Il vous faudra parfois mettre votre système d'aide à jour pour obtenir ces informations.

Utilisation du composant Label

Utilisez des composants Label afin de créer des étiquette de texte pour les autres composants de formulaire, tels que l'étiquette « Nom : ». placée à gauche d'un champ TextInput où l'on saisirait un nom d'utilisateur. Si vous créez une application à l'aide de composants basés sur la version 2 (v2) de Macromedia Component Architecture, il est préférable d'utiliser un composant Label au lieu d'un champ de texte ordinaire afin d'employer des styles qui vous permettront de conserver un aspect cohérent dans l'ensemble du document.

Paramètres Label

Vous trouverez ci-dessous les paramètres de création qu'il est possible de définir dans le panneau de l'inspecteur des propriétés ou des composants pour toutes les occurrences des composants Label :

text indique le texte de l'étiquette ; la valeur par défaut est Etiquette.

html indique si l'étiquette est formatée avec HTML (`true`) ou non (`false`). Si le paramètre `html` est défini sur `true`, aucun style ne peut être défini pour l'étiquette. La valeur par défaut est `false`.

autoSize indique les dimensions de l'étiquette et son alignement par rapport au texte. La valeur par défaut est none. Ce paramètre peut avoir l'une des quatre valeurs suivantes :

- none—l'étiquette n'est pas redimensionnée ou alignée pour contenir le texte.
- left—le bas et le côté droit de l'étiquette sont redimensionnés pour contenir le texte. Le haut et le côté gauche ne sont pas redimensionnés.
- center—le bas de l'étiquette est redimensionné pour contenir le texte. Le centre horizontal de l'étiquette reste ancré à sa position d'origine.
- right—le bas et le côté gauche de l'étiquette sont redimensionnés pour contenir le texte. Le haut et le côté droit ne sont pas redimensionnés.

Remarque : La propriété `autoSize` du composant Label est différente de la propriété `autoSize` intégrée à l'objet ActionScript TextField.

Vous pouvez rédiger des instructions ActionScript afin de définir d'autres options pour les occurrences d'étiquette à l'aide des méthodes, des propriétés et des événements. Pour plus d'informations, consultez Classe Label.

Création d'une application avec le composant Label

La procédure suivante explique comment ajouter un composant Label à une application au cours de la programmation. Dans cet exemple, l'étiquette se trouve à côté d'une liste déroulante contenant des dates, dans une application de panier d'achat.

Pour créer une application avec le composant Label, procédez comme suit :

1 Faites glisser un composant Label du panneau Composants jusqu'à la scène.

2 Dans le panneau Inspecteur de composants, procédez comme suit :

- Saisissez une **date d'expiration** pour le paramètre de l'étiquette.

Personnalisation du composant Label

Vous pouvez orienter un composant Label horizontalement et verticalement au cours de la programmation et lors de l'exécution. Lors de la création, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. Vous pouvez également définir le paramètre de création autoSize ; la définition de ce paramètre ne change pas le cadre de délimitation dans l'aperçu en direct, mais l'étiquette est redimensionnée. Pour plus d'informations, consultez *Paramètres Label*, page 111. Lors de l'exécution, utilisez la méthode `setSize()` (consultez `UIObject.setSize()`) ou `Label.autoSize`.

Utilisation des styles avec le composant Label

Vous pouvez définir des propriétés de style afin de modifier l'aspect d'une occurrence d'étiquette. Tout le texte contenu dans une occurrence de composant Label doit partager le même style. Par exemple, vous ne pouvez pas définir le style de `color` sur `"blue"` pour un mot de l'étiquette et le définir sur `"red"` pour un autre mot de la même étiquette.

Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style.

Pour plus d'informations sur les styles, consultez *Utilisation des styles pour personnaliser la couleur et le texte des composants*, page 27.

Les composants Label supportent les styles suivants :

Style	Description
`color`	Couleur par défaut pour le texte.
`embedFonts`	Polices à incorporer dans le document.
`fontFamily`	Nom de police pour du texte.
`fontSize`	Taille en points pour la police.
`fontStyle`	Style de la police, "normal" ou "italic".
`fontWeight`	Epaisseur de la police, 'normal" ou "bold".
`textAlign`	Alignement du texte : "left", "right" ou "center".
`textDecoration`	Décoration du texte, "none" ou "underline".

Utilisation des enveloppes avec le composant Label

Aucune enveloppe ne peut être appliquée au composant Label.

Pour plus d'informations sur l'application d'une enveloppe à un composant, consultez *A propos de l'application des enveloppes aux composants*, page 37.

Classe Label

Héritage UIObject > Label

Espace de nom de classe ActionScript mx.controls.Label

Lors de l'exécution, les propriétés de la classe Label permettent de spécifier du texte pour l'étiquette, d'indiquer si le texte peut être formaté avec HTML et de spécifier si l'étiquette sera automatiquement dimensionnée en fonction de la taille du texte.

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

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

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

Remarque : Le code suivant renvoie la valeur undefined : `trace(monOccurrenceEtiquette.version);`.

Méthodes de la classe Label

Hérite de toutes les méthodes de la Classe UIObject.

Propriétés de la classe Label

Propriété	Description
`Label.autoSize`	Chaîne qui indique la manière dont une étiquette sera dimensionnée et alignée sur la valeur de sa propriété `text`. Quatre valeurs sont possibles : `"none"`, `"left"`, `"center"` et `"right"`. La valeur par défaut est `"none"`.
`Label.html`	Une valeur booléenne indiquant si une étiquette peut être formatée avec HTML (`true`) ou non (`false`).
`Label.text`	Texte de l'étiquette

Hérite de toutes les propriétés de la Classe UIObject.

Evénements de la classe Label

Hérite de tous les événements de la Classe UIObject.

Label.autoSize

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`occurrenceEtiquette``.autoSize`

Description

Propriété : chaîne indiquant la manière dont une étiquette sera dimensionnée et s'alignera sur la valeur de sa propriété `text`. Quatre valeurs sont possibles : `"none"`, `"left"`, `"center"` et `"right"`. La valeur par défaut est `"none"`.

- `none`—l'étiquette n'est pas redimensionnée ou alignée pour contenir le texte.
- `left`—le bas et le côté droit de l'étiquette sont redimensionnés pour contenir le texte. Le haut et le côté gauche ne sont pas redimensionnés.

- `center`—le bas de l'étiquette est redimensionné pour contenir le texte. Le centre horizontal de l'étiquette reste ancré à sa position d'origine.
- `right`—le bas et le côté gauche de l'étiquette sont redimensionnés pour contenir le texte. Le haut et le côté droit ne sont pas redimensionnés.

Remarque : La propriété `autoSize` du composant Label est différente de la propriété `autoSize` intégrée à l'objet ActionScript TextField.

Label.html

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceEtiquette.html

Description

Propriété : valeur booléenne indiquant si l'étiquette peut être formatée avec HTML (`true`) ou non (`false`). La valeur par défaut est `false`. Le style des composants Label ayant la propriété `html` définie sur `true` ne peut pas être formaté.

Vous ne pouvez pas utiliser la balise HTML `<font color>` avec le composant Label, même si `Label.html` est défini sur `true`. Dans l'exemple suivant, le texte « Bonjour » est affiché en noir et non en rouge comme il le serait si `<font color>` était supporté :

```
lbl.html = true;
lbl.text = "<font color=\"#FF0000\">Bonjour</font> le monde";
```

Pour extraire le texte ordinaire d'un texte au format HTML, définissez la propriété `HTML` sur `false`, puis accédez à la propriété `text`. Cette procédure ayant pour effet de supprimer le formatage HTML, il peut être souhaitable, avant de l'exécuter, de copier le texte de l'étiquette dans un composant Label ou TextArea hors écran.

Exemple

L'exemple suivant définit la propriété `html` sur `true` pour que l'étiquette puisse être formatée avec HTML. La propriété `text` est ensuite définie sur une chaîne incluant du formatage HTML, comme dans l'exemple suivant :

```
labelControl.html = true;
labelControl.text = "Le blabla <b>royal</b>";
```

Le mot « royal » est affiché en caractères gras.

Label.text

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceEtiquette.text

Description

Propriété : texte d'une étiquette. La valeur par défaut est `"Label"`.

Exemple

Le code suivant définit la propriété `text` de l'occurrence d'étiquette `contrôleEtiquette` et envoie la valeur au panneau de sortie :

```
contrôleEtiquette.text = "Le blabla royal";
trace(labelControl.text);
```

Composant List

Le composant List est une zone de liste défilante à sélection unique ou multiple. Les listes peuvent également contenir des graphiques, y compris d'autres composants. L'ajout des éléments affichés dans la zone de liste s'effectue via la boîte de dialogue Valeurs qui s'affiche lorsque vous cliquez dans les champs de paramètres des étiquettes ou des données. Vous pouvez également utiliser les méthodes `List.addItem()` et `List.addItemAt()` pour ajouter des éléments à la liste.

Le composant List utilise un index basé sur zéro, où l'élément possédant l'index 0 est le premier affiché. Lorsque vous ajoutez, supprimez ou remplacez les éléments d'une liste au moyen des méthodes de la classe List, il peut s'avérer nécessaire de définir l'index de ces éléments.

La liste reçoit le focus lorsque vous cliquez dessus ou appuyez sur la touche Tab pour y accéder. Vous pouvez utiliser les touches suivantes pour la contrôler :

Touche	Description
Touches alphanumériques	Passe à l'élément suivant avec l'étiquette Key.getAscii() en tant que premier caractère de l'étiquette.
Contrôle	Bouton bascule. Permet plusieurs sélections et désélections non contiguës.
Bas	La sélection se déplace d'un élément vers le bas.
Origine	La sélection se déplace jusqu'au sommet de la liste.
Pg. Suiv.	La sélection se déplace sur la page suivante.
Pg. Préc.	La sélection se déplace sur la page précédente.
Maj	Touche de sélection contiguë. Permet une sélection contiguë.
Haut	La sélection se déplace d'un élément vers le haut.

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

Pour plus d'informations sur le contrôle du focus, consultez *Création de la navigation personnalisée du focus*, page 25 ou *Classe FocusManager*, page 103.

Un aperçu en direct de chaque occurrence de liste sur la scène reflète les modifications apportées à leurs paramètres dans le panneau de l'inspecteur des propriétés ou des composants au cours de la programmation.

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

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

Vous n'activez l'accessibilité d'un composant qu'une fois, quel que soit son nombre d'occurrences. Pour plus d'informations, consultez « Création de contenu accessible », dans le guide Utilisation de Flash de l'aide. Il vous faudra parfois mettre votre système d'aide à jour pour obtenir ces informations.

Utilisation du composant List

Vous pouvez définir une liste dans laquelle les utilisateurs pourront effectuer un ou plusieurs choix. Par exemple, un utilisateur qui visite un site web de commerce électronique a besoin de choisir l'article à acheter. Il y a ici 30 articles. L'utilisateur fait défiler la liste et en choisit un en cliquant dessus.

Vous pouvez également utiliser des clips personnalisés en tant que lignes dans la liste. Ils vous permettront de donner des informations supplémentaires aux utilisateurs. Par exemple, dans une application de courrier électronique, toutes les boîtes de réception pourraient être des composants List et toutes les lignes pourraient être accompagnées d'icônes indiquant leur priorité et leur état.

Paramètres du composant List

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

data Tableau de valeurs qui constituent les données de la liste. La valeur par défaut est [] (tableau vide). Il n'existe pas de propriété équivalente lors de l'exécution.

labels Tableau des valeurs de texte qui remplissent les valeurs des étiquettes de la liste. La valeur par défaut est [] (tableau vide). Il n'existe pas de propriété équivalente lors de l'exécution.

multipleSelection Valeur booléenne qui indique si vous pouvez sélectionner plusieurs valeurs (true) ou non (false). La valeur par défaut est false.

rowHeight indique la hauteur, en pixels, de chaque ligne. La valeur par défaut est 20. La définition d'une police ne change rien à la hauteur de la ligne.

Vous pouvez rédiger des instructions ActionScript afin de définir d'autres options pour les occurrences de liste à l'aide des méthodes, des propriétés et des événements. Pour plus d'informations, consultez Classe List.

Création d'une application avec le composant List

La procédure suivante explique comment ajouter un composant List à une application au cours de la programmation. Dans cet exemple, la liste est un échantillon composé de trois éléments.

Pour ajouter un composant List simple à une application, procédez comme suit :

1 Faites glisser un composant List du panneau Composants jusqu'à la scène.

2 Sélectionnez la liste et choisissez Modification > Transformer pour l'adapter aux dimensions de votre application.

3 Dans l'inspecteur des propriétés, procédez comme suit :

- Saisissez le nom d'occurrence **maListe**.
- Saisissez Elément1, Elément2 et Elément3 pour les paramètres des étiquettes.
- Saisissez élément1.html, élément2.html, élément3.html pour les paramètres de données.

4 Choisissez Contrôle > Tester l'animation pour visualiser la liste et ses éléments.

Vous pouvez utiliser les valeurs de propriété de données dans votre application pour ouvrir les fichiers HTML.

La procédure suivante explique comment ajouter un composant List à une application au cours de la programmation. Dans cet exemple, la liste est un échantillon composé de trois éléments.

Pour ajouter un composant List complexe à une application, procédez comme suit :

1 Faites glisser un composant List du panneau Composants jusqu'à la scène.

2 Sélectionnez la liste et choisissez Modification > Transformer pour l'adapter aux dimensions de votre application.

3 Dans le panneau Actions, saisissez le nom d'occurrence **maListe.**

4 Sélectionnez l'image 1 du scénario et, dans le panneau Actions, saisissez ce qui suit :

```
maListe.dataProvider = monFD;
```

Si vous avez déterminé un fournisseur de données intitulé `monFD`, la liste se remplit de données. Pour plus d'informations sur les fournisseurs de données, consultez `List.dataProvider`.

5 Choisissez Contrôle > Tester l'animation pour visualiser la liste et ses éléments.

Personnalisation du composant List

Vous pouvez orienter un composant List horizontalement et verticalement pendant la création et l'exécution. Lors de la création, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez la méthode `List.setSize()` (consultez `UIObject.setSize()`).

Lorsqu'une liste est redimensionnée, ses lignes diminuent horizontalement, ce qui coupe le texte qui se trouve à l'intérieur. Verticalement, la liste ajoute ou supprime le nombre de lignes approprié. Les barres de défilement se placent automatiquement. Pour plus d'informations sur les barres de défilement, consultez *Composant ScrollPane*, page 190.

Utilisation des styles avec le composant List

Vous pouvez définir des propriétés de style afin de modifier l'aspect d'un composant List.

Un composant List utilise les styles de halo suivants :

Style	Description
`alternatingRowColors`	Spécifie des couleurs alternées pour les lignes. La valeur peut être un tableau de couleurs, 0xFF00FF, 0xCC6699 et 0x996699, par exemple.
`backgroundColor`	Couleur d'arrière-plan de la liste. Ce style est défini sur une déclaration de style de classe, ScrollSelectList.

Style	Description
borderColor	Section noire d'une bordure à trois dimensions ou section de couleur d'une bordure à deux dimensions.
borderStyle	Style du cadre de délimitation. Les valeurs possibles sont : "none", "solid", "inset" et "outset". Ce style est défini sur une déclaration de style de classe, ScrollSelectList.
defaultIcon	Nom de l'icône à utiliser par défaut pour les lignes de la liste. La valeur par défaut est undefined.
rollOverColor	Couleur d'une ligne survolée.
selectionColor	Couleur d'une ligne sélectionnée.
selectionEasing	Référence à une équation d'accélération (fonction) utilisée pour contrôler l'interpolation de programmation.
disabledColor	Couleur désactivée pour du texte.
textRollOverColor	Couleur du texte lorsque le pointeur passe dessus.
textSelectedColor	Couleur du texte sélectionné.
selectionDisabledColor	Couleur d'une ligne sélectionnée et désactivée.
selectionDuration	Longueur des transitions lors de la sélection des éléments.
useRollOver	Détermine si le survol d'une ligne active sa mise en surbrillance.

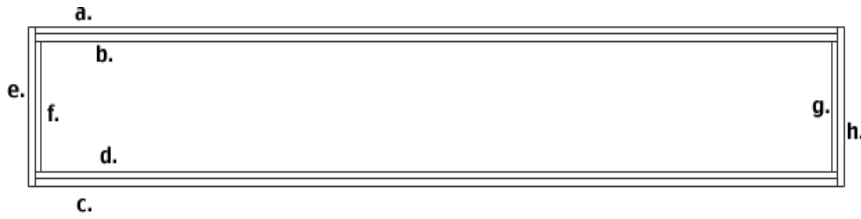
Les composants List utilisent également les propriétés de style des composants Label (consultez *Utilisation des styles avec le composant Label*, page 112), ScrollBar et RectBorder.

Utilisation des enveloppes avec le composant List

Toutes les enveloppes du composant List sont incluses dans les sous-composants qui constituent la liste (Composant ScrollPane et RectBorder). Pour plus d'informations, consultez *Composant ScrollPane*, page 190. La méthode `setStyle()` (consultez `UIObject.setStyle()`) vous permet de modifier les propriétés suivantes du style RectBorder :

Styles RectBorder	Position de la bordure
borderColor	a
highlightColor	b
borderColor	c
shadowColor	d
borderCapColor	e
shadowCapColor	f
shadowCapColor	g
borderCapColor	h

Les propriétés de style définissent les positions suivantes sur la bordure :



Classe List

Héritage UIObject > UIComponent > View > ScrollView > ScrollSelectList > List

Espace de nom de classe ActionScript mx.controls.List

Le composant List est constitué de trois parties :

- Eléments
- Lignes
- Fournisseur de données

Un élément est un objet ActionScript utilisé pour stocker les unités d'informations dans la liste. Une liste peut être représentée un peu comme un tableau ; chaque espace indexé du tableau est un élément. Un élément est un objet disposant, en règle générale, d'une propriété `label` qui est affichée et d'une propriété `data` qui sert à stocker des données.

Une ligne est un composant utilisé pour afficher un élément. Les lignes sont fournies par défaut par la liste (la classe SelectableRow est utilisée) ou vous pouvez les fournir, en général sous la forme de sous-classes de la classe SelectableRow. La classe SelectableRow implémente l'interface CellRenderer, ensemble des propriétés et méthodes qui permettent à la liste de manipuler toutes les lignes et d'envoyer des données et des informations d'état (par exemple, mises en surbrillance, sélections, etc.) à la ligne affichée.

Le fournisseur de données correspond au modèle des données des éléments dans une liste. Un tableau situé dans la même image qu'une liste reçoit automatiquement des méthodes qui permettent de manipuler les données et de diffuser les changements vers plusieurs affichages. Vous pouvez créer une occurrence de tableau ou en obtenir une d'un serveur et l'utiliser comme modèle de données pour plusieurs composants List, ComboBox, DataGrid, etc. Le composant List a un jeu de méthodes qui agit comme proxy pour le fournisseur de données (par exemple, `addItem()` et `removeItem()`). Si aucun fournisseur de données externe n'est fourni à la liste, ces méthodes créent automatiquement une occurrence de fournisseur de données, exposée par le biais de `List.dataProvider`.

Pour ajouter un composant List à l'ordre des tabulations d'une application, définissez sa propriété `tabIndex` (consultez `UIComponent.tabIndex`). Le composant List utilise FocusManager pour remplacer le rectangle de focus par défaut de Flash Player et tracer un rectangle de focus personnalisé aux coins arrondis. Pour plus d'informations, consultez *Création de la navigation personnalisée du focus*, page 25.

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

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

Remarque : Le code suivant renvoie la valeur undefined : `trace(OccurrenceMaListe.version);`.

Résumé des méthodes de la classe List

Méthode	Description
`List.addItem()`	Ajoute un élément à la fin de la liste.
`List.addItemAt()`	Ajoute un élément à la liste, à l'index spécifié.
`List.getItemAt()`	Renvoie l'élément à l'emplacement d'index spécifié.
`List.removeAll()`	Supprime tous les éléments de la liste.
`List.removeItemAt()`	Supprime l'élément à l'index spécifié.
`List.replaceItemAt()`	Remplace l'élément, à l'index spécifié, par un autre élément.
`List.setPropertiesAt()`	Applique les propriétés spécifiées à un élément donné.
`List.sortItems()`	Trie les éléments de la liste à l'aide de la fonction de comparaison spécifiée.
`List.sortItemsBy()`	Trie les éléments de la liste à l'aide d'une propriété donnée.

Hérite de toutes les méthodes de la Classe UIObject et de la Classe UIComponent.

Propriétés de la classe List

Propriété	Description
`List.cellRenderer`	Affecte l'objet cellRenderer à utiliser pour chaque ligne de la liste.
`List.dataProvider`	Source des éléments de la liste.
`List.hPosition`	Position horizontale de la liste.
`List.hScrollPolicy`	Indique si la barre de défilement horizontale est affichée (`"on"`) ou non (`"off"`).
`List.iconField`	Champ situé à l'intérieur de chaque élément et utilisé pour spécifier les icônes.
`List.iconFunction`	Fonction qui détermine les icônes à utiliser.
`List.labelField`	Spécifie un champ dans chaque élément, à utiliser comme texte d'étiquette.
`List.labelFunction`	Fonction qui détermine les champs de chaque élément à utiliser pour le texte d'étiquette.
`List.length`	Longueur de la liste en éléments. Cette propriété est en lecture seule.

Propriété	Description
List.maxHPosition	Spécifie le nombre de pixels que la liste peut faire défiler à droite, lorsque List.hScrollPolicy est défini sur "on".
List.multipleSelection	Indique si la sélection multiple est autorisée dans la liste (true) ou non (false).
List.rowCount	Nombre de lignes au moins partiellement visibles dans la liste.
List.rowHeight	Hauteur des pixels de chaque ligne de la liste.
List.selectable	Indique si la liste peut être sélectionnée (true) ou non (false).
List.selectedIndex	Index d'une sélection dans une liste à sélection unique.
List.selectedIndices	Tableau des éléments sélectionnés dans une liste à sélection multiple.
List.selectedItem	Elément sélectionné dans une liste à sélection unique. Cette propriété est en lecture seule.
List.selectedItems	Objets sélectionnés dans une liste à sélection multiple. Cette propriété est en lecture seule.
List.vPosition	Fait défiler la liste pour que le premier élément visible soit le numéro affecté.
List.vScrollPolicy	Indique si la barre de défilement verticale est affichée ("on"), pas affichée ("off") ou affichée si nécessaire ("auto").

Hérite de toutes les propriétés de la Classe UIObject et de la Classe UIComponent.

Evénements de la classe List

Evénement	Description
List.change	Diffusé chaque fois que la sélection change suite à une interaction avec l'utilisateur.
List.itemRollOut	Diffusé lorsque les éléments de la liste sont survolés par le pointeur et à la sortie du survol.
List.itemRollOver	Diffusé lorsque les éléments de la liste sont survolés par le pointeur.
List.scroll	Diffusé lorsqu'une liste défile.

Hérite de tous les événements de la Classe UIObject et de la Classe UIComponent.

List.addItem()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceListe.addItem(*label*[, *data*])

OccurrenceListe.addItem(*objetElément*)

Paramètres

étiquette Chaîne indiquant l'étiquette du nouvel élément.

données Données de l'élément. Ce paramètre est facultatif et peut correspondre à n'importe quel type de données.

objetElément Objet d'élément possédant généralement des propriétés `label` et `data`.

Renvoie

L'index auquel l'élément est ajouté.

Description

Méthode : ajoute un nouvel élément à la fin de la liste.

Dans le premier exemple d'utilisation, un objet est toujours créé avec la propriété `label` spécifiée et de la propriété `data`, le cas échéant.

Le deuxième exemple d'utilisation ajoute l'objet spécifié.

L'appel de cette méthode modifie le fournisseur de données du composant List. Si le fournisseur de données est partagé par d'autres composants, ceux-ci sont également mis à jour.

Exemple

Les deux lignes de code suivantes ajoutent un élément à l'occurrence `maListe`. Pour essayer ce code, faites glisser une liste jusqu'à la scène et donnez-lui le nom d'occurrence **maListe**. Ajoutez le code suivant à l'Image 1 du scénario :

```
maListe.addItem("ceci est un élément");
maListe.addItem({label:"Gabriel",age:"très âgé",data:123});
```

List.addItemAt()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceListe.*addItemAt*(*index*, *label* [,*data*])

OccurrenceListe.addItemAt(*index*, *objetElément*)

Paramètres

étiquette Chaîne indiquant l'étiquette du nouvel élément.

données Données de l'élément. Ce paramètre est facultatif et peut correspondre à n'importe quel type de données.

index Nombre supérieur ou égal à zéro qui indique la position de l'élément.

objetElément Objet d'élément possédant généralement des propriétés `label` et `data`.

Renvoie

L'index auquel l'élément est ajouté.

Description

Méthode ; ajoute un nouvel élément à la position spécifiée par le paramètre *index*.

Dans le premier exemple d'utilisation, un objet est toujours créé avec la propriété `label` spécifiée et de la propriété `data`, le cas échéant.

Le deuxième exemple d'utilisation ajoute l'objet spécifié.

L'appel de cette méthode modifie le fournisseur de données du composant List. Si le fournisseur de données est partagé par d'autres composants, ceux-ci sont également mis à jour.

Exemple

La ligne de code suivante ajoute un élément à la troisième position d'index, qui correspond au quatrième élément dans la liste :

```
maListe.addItemAt(3,{label:'Red',data:0xFF0000});
```

List.cellRenderer

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceListe`.cellRenderer`

Description

Propriété : affecte l'objet cellRenderer à utiliser pour chaque ligne de la liste. Cette propriété doit correspondre à une référence d'objet de classe ou à un identificateur de liaison de symbole pour que l'objet cellRenderer puisse l'utiliser. Toutes les classes utilisées pour cette propriété doivent implémenter *Interface CellRenderer*, page 61.

Exemple

L'exemple suivant utilise un identificateur de liaison pour définir un nouvel objet cellRenderer :

```
maListe.cellRenderer = "ComboBoxCell";
```

List.change

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

```
listenerObject = new Object();
objetDécoute.change = function(objetEvnt){
  // votre code ici
}
OccurrenceListe.addEventListener("change", objetDécoute)
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque l'index sélectionné dans la liste change suite à l'interaction d'un utilisateur.

Le premier exemple d'utilisation utilise un gestionnaire `on()` et doit être directement lié à une occurrence de composant List. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de comportement. Par exemple, le code suivant, lié à l'occurrence de composant List `monChamp`, envoie « _level0.monChamp » au panneau Sortie :

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`OccurrenceListe`*) distribue un événement (ici, `change`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Pour finir, vous appelez la méthode `addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur avec l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Exemple

L'exemple suivant envoie le nom de l'occurrence du composant ayant généré l'événement `change` au panneau de sortie :

```
form.change = function(objEvnt){
  trace("Valeur passée à " + objEvnt.target.value);
}
maListe.addEventListener("change", form);
```

Consultez également

`UIEventDispatcher.addEventListener()`

List.dataProvider

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
OccurrenceListe.dataProvider
```

Description

Propriété : modèle de données pour les éléments affichés dans une liste. La valeur de cette propriété peut être un tableau ou n'importe quel objet mettant en œuvre l'interface DataProvider. La valeur par défaut est []. Pour plus d'informations sur l'interface DataProvider, consultez *Composant DataProvider*, page 97.

Le composant List et les autres composants de données ajoutent des méthodes au prototype de l'objet Tableau pour être conformes à l'interface DataProvider. Tout tableau existant aussi en tant que liste dispose donc automatiquement de toutes les méthodes (`addItem()`, `getItemAt()`, etc.) nécessaires pour être le modèle de données de la liste et peut être utilisé pour diffuser les changements de modèle à plusieurs composants.

Si le tableau contient des objets, l'utilisateur accède aux propriétés `List.labelField` ou `List.labelFunction` pour déterminer les parties de l'élément à afficher. La valeur par défaut est « `label` », ce qui signifie que si un champ `label` existe, il est affiché. S'il n'existe pas, une liste de tous les champs séparés par une virgule est affichée.

Remarque : Si le tableau contient des chaînes et aucun objet à tous les emplacements d'index, la liste ne peut pas trier les éléments et conserver l'état de sélection. Le tri entraîne la perte de la sélection.

Toute occurrence implémentant l'interface DataProvider peut agir comme un fournisseur de données pour un composant List. Cela inclut Flash Remoting RecordSets, Firefly DataSets, etc.

Exemple

Cet exemple utilise un tableau de chaînes pour remplir la liste :

```
list.dataProvider = ["Expédition terrestre","Surlendemain, par
  avion","Lendemain, par avion"];
```

Cet exemple crée un tableau fournisseur de données et lui affecte la propriété `dataProvider`, comme suit :

```
monFD = new Array();
list.dataProvider = monFD;

pour (var i=0; i<accounts.length; i++) {
  // ces modifications apportées à DataProvider seront diffusées dans la liste
  monFD.addItem({ label: accounts[i].name,
                  data:  accounts[i].accountID });
}
```

List.getItemAt()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceListe.getItemAt(*index*)

Paramètres

index Nombre supérieur ou égal à 0, et inférieur à `List.length`. Index de l'élément à récupérer.

Renvoie

Objet d'élément indexé. Non défini si l'index est en dehors des limites.

Description

Méthode : récupère l'élément à l'emplacement d'index spécifié.

Exemple

Le code suivant affiche l'étiquette à l'emplacement d'index 4 :

```
trace(maListe.getItemAt(4).label);
```

List.hPosition

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceListe.hPosition

Description

Propriété : fait défiler la liste horizontalement en fonction du nombre de pixels spécifié. Il est impossible de définir `hPosition` à moins que la valeur de `hScrollPolicy` soit réglée sur `"on"` et que la valeur de `maxHPosition` soit supérieure à 0.

Exemple

L'exemple suivant détermine la position de défilement horizontal de `maListe` :

```
var scrollPos = maListe.hPosition;
```

L'exemple suivant définit la position de défilement horizontal sur tout le côté gauche :

```
maListe.hPosition = 0;
```

List.hScrollPolicy

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceListe.hScrollPolicy

Description

Propriété : chaîne qui détermine si la barre de défilement horizontale est affichée ou non ; la valeur peut être "on" ou "off". La valeur par défaut est "off". La barre de défilement horizontale ne mesure pas le texte. Vous devez définir une position de défilement horizontale maximale. Consultez List.maxHPosition.

Remarque : La valeur "auto" n'est pas prise en charge pour List.hScrollPolicy.

Exemple

Le code suivant permet à la liste de défiler horizontalement jusqu'à 200 pixels :

```
maListe.hScrollPolicy = "on";
maListe.Box.maxHPosition = 200;
```

Consultez également

List.hPosition, List.maxHPosition

List.iconField

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceListe.iconField

Description

Propriété : spécifie le nom d'un champ à utiliser en tant qu'identificateur d'icône. Si la valeur du champ n'est pas définie, l'icône par défaut spécifiée par le style defaultIcon est utilisée. Si le style defaultIcon n'est pas défini, aucune icône n'est utilisée.

Exemple

L'exemple suivant définit la propriété iconField sur la propriété icon de chaque élément :

```
list.iconField = "icon"
```

Consultez également

List.iconFunction

List.iconFunction

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
OccurrenceListe.iconFunction
```

Description

Propriété : spécifie une fonction à utiliser pour déterminer quelle icône sera utilisée pour afficher l'élément de chacune des lignes. Cette fonction reçoit un paramètre, *item*, qui correspond à l'élément présenté, et doit renvoyer une chaîne représentant l'identificateur de symbole de l'icône.

Exemple

L'exemple suivant ajoute des icônes qui indiquent si un fichier correspond à un document image ou texte. Si le champ `data.fileExtension` contient une valeur `"jpg"` ou `"gif"`, l'icône utilisée sera `"pictureIcon"`, etc. :

```
list.iconFunction = function(item){
  var type = item.data.fileExtension;
  if (type=="jpg" || type=="gif") {
    return "pictureIcon";
  } else if (type=="doc" || type=="txt") {
    return "docIcon";
  }
}
```

List.itemRollOut

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(itemRollOut){
  // votre code ici
}
```

Usage 2 :

```
listenerObject = new Object();
objetDécoute.itemRollOut = function(ObjetEvnt){
  // votre code ici
}
OccurrenceListe.addEventListener("itemRollOut", objetDécoute)
```

Objet événement

Outre les propriétés standard de l'objet événement, l'événement `itemRollOut` possède une propriété supplémentaire : `index`. L'`index` correspond au numéro de l'élément à la sortie du survol.

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque les éléments de la liste sont survolés.

Le premier exemple d'utilisation utilise un gestionnaire `on()` et doit être directement lié à une occurrence de composant List. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de comportement. Par exemple, le code suivant, lié à l'occurrence de composant List `maListe`, envoie « _level0.maListe » au panneau Sortie :

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`OccurrenceListe`*) distribue un événement (ici, `itemRollOut`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

L'exemple suivant envoie un message au panneau de sortie indiquant le numéro d'index survolé par le pointeur :

```
form.itemRollOut = function (objEvnt) {
  trace("Item #" + objetEvnt.index + " a été survolé.");
}
maListe.addEventListener("itemRollOut", form);
```

Consultez également

`List.itemRollOver`

List.itemRollOver

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(itemRollOver){
  // votre code ici
}
```

Usage 2 :

```
listenerObject = new Object();
objetDécoute.itemRollOver = function(ObjetEvnt){
  // votre code ici
}
OccurrenceListe.addEventListener("itemRollOver", objetDécoute)
```

Objet événement

Outre les propriétés standard de l'objet événement, l'événement `itemRollOver` possède une propriété supplémentaire : `index`. L'`index` est le numéro de l'élément survolé par le pointeur.

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque les éléments de la liste sont survolés.

Le premier exemple d'utilisation utilise un gestionnaire `on()` et doit être directement lié à une occurrence de composant List. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de comportement. Par exemple, le code suivant, lié à l'occurrence de composant List `maListe`, envoie « _level0.maListe » au panneau Sortie :

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`OccurrenceListe`*) distribue un événement (ici, `itemRollOver`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

L'exemple suivant envoie un message au panneau de sortie indiquant le numéro d'index survolé par le pointeur :

```
form.itemRollOver = function (objEvnt) {
  trace("Item #" + objEvnt.index + " survolé.");
}
maListe.addEventListener("itemRollOver", form);
```

Consultez également

`List.itemRollOut`

List.labelField

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceListe.labelField

Description

Propriété : spécifie, dans chaque élément, un champ à utiliser comme texte d'affichage. Cette propriété prend la valeur du champ et l'utilise comme étiquette. La valeur par défaut est "étiquette".

Exemple

L'exemple suivant définit la propriété labelField sur le champ "nom" de chaque élément. L'élément ajouté à la deuxième ligne de code aurait "Nina" comme étiquette :

```
list.labelField = "nom";
list.addItem({name: "Nina", age: 25});
```

Consultez également

List.labelFunction

List.labelFunction

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceListe.labelFunction

Description

Propriété : spécifie une fonction à utiliser pour décider du champ (ou de la combinaison de champs) à afficher. Cette fonction reçoit un paramètre, *item*, qui correspond à l'élément présenté, et doit renvoyer une chaîne représentant le texte à afficher.

Exemple

L'exemple suivant affiche des détails formatés des éléments dans l'étiquette :

```
list.labelFunction = function(item){
  return "Le prix du produit " + item.productID + ", " + item.productName + "
  est $"
  + item.price;
}
```

Consultez également

List.labelField

List.length

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceListe.length

Description

Propriété (lecture seule) : nombre d'éléments dans la liste.

Exemple

L'exemple suivant place la valeur de `length` dans une variable :

```
var len = maListe.length;
```

List.maxHPosition

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceListe.maxHPosition

Description

Propriété : spécifie le nombre de pixels que la liste peut faire défiler, lorsque List.hScrollPolicy est défini sur `"on"`. La liste ne mesure pas précisément la largeur du texte qu'elle contient. Vous devez définir `maxHPosition` pour indiquer le volume de défilement requis. Si cette propriété n'est pas définie, la liste ne défile pas horizontalement.

Exemple

L'exemple suivant crée une liste avec 400 pixels de défilement horizontal :

```
maListe.hScrollPolicy = "on";
maListe.maxHPosition = 400;
```

Consultez également

List.hScrollPolicy

List.multipleSelection

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceListe.multipleSelection

Description

Propriété : indique si les sélections multiples sont autorisées (`true`) ou si seules les sélections uniques sont autorisées (`false`). La valeur par défaut est `false`.

Exemple

L'exemple suivant effectue un test afin de déterminer si plusieurs éléments peuvent être sélectionnés :

```
if (maListe.multipleSelection){
  // votre code ici
}
```

L'exemple suivant permet les sélections multiples dans la liste :

```
maListe.selectMultiple = true;
```

List.removeAll()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceListe.removeAll()

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : supprime tous les éléments de la liste.

L'appel de cette méthode modifie le fournisseur de données du composant List. Si le fournisseur de données est partagé par d'autres composants, ceux-ci sont également mis à jour.

Exemple

Le code suivant supprime tous les éléments de la liste :

```
maListe.removeAll();
```

List.removeItemAt()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
occurrenceListe.removeItemAt(index)
```

Paramètres

index Chaîne qui indique l'étiquette utilisée pour le nouvel élément. Valeur supérieure à zéro et inférieure à `List.length`.

Renvoie

Un objet : l'élément supprimé (non défini si aucun élément n'existe).

Description

Méthode : supprime un élément à l'emplacement d'*index* indiqué. Un index disparaît parmi les index de la liste situés après l'index indiqué par le paramètre *index*.

L'appel de cette méthode modifie le fournisseur de données du composant List. Si le fournisseur de données est partagé par d'autres composants, ceux-ci sont également mis à jour.

Exemple

Le code suivant supprime l'élément à l'emplacement d'index 3 :

```
maListe.removeItemAt(3);
```

List.replaceItemAt()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
OccurrenceListe.replaceItemAt(index, label[, data])
OccurrenceListe.replaceItemAt(index, objetElément)
```

Paramètres

index Nombre supérieur à zéro et inférieur à `List.length` indiquant la position à laquelle insérer l'élément (index du nouvel élément).

étiquette Chaîne indiquant l'étiquette du nouvel élément.

données Données de l'élément. Ce paramètre est facultatif est peut être de n'importe quel type.

objetElément. Objet à utiliser en tant qu'élément. Il possède généralement les propriétés `label` et `data`.

Renvoie

Rien.

Description

Méthode : remplace le contenu de l'élément à l'emplacement d'index spécifié par le paramètre *index*.

L'appel de cette méthode modifie le fournisseur de données du composant List. Si le fournisseur de données est partagé par d'autres composants, ceux-ci sont également mis à jour.

Exemple

L'exemple suivant modifie la quatrième position d'index :

```
maListe.replaceItemAt(3, "nouvelle étiquette");
```

List.rowCount

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceListe`.rowCount`

Description

Propriété : le nombre de lignes au moins partiellement visibles dans la liste. Ceci s'avère utile si vous avez dimensionné une liste en pixels et devez compter ses lignes. Inversement, la définition du nombre de lignes garanti qu'un nombre exact de lignes sera affiché, sans ligne partielle en bas.

Le code `maListe.rowCount = num` équivaut au code `maListe.setSize(maListe.width, h)` (où `h` est la hauteur requise pour afficher les éléments `num`).

La valeur par défaut est fondée sur la hauteur de la liste, telle qu'elle est définie au cours de la programmation, ou par la méthode `list.setSize()` (consultez `UIObject.setSize()`).

Exemple

L'exemple suivant montre le nombre d'éléments visibles dans une liste :

```
var rowCount = maListe.rowCount;
```

L'exemple suivant permet d'afficher quatre éléments dans la liste :

```
maListe.rowCount = 4;
```

Cet exemple supprime une ligne partielle en bas d'une liste, le cas échéant :

```
maListe.rowCount = maListe.rowCount;
```

Cet exemple définit une liste au plus petit nombre de lignes qu'elle peut afficher entièrement.

```
maListe.rowCount = 1;
trace("maListe contient "+maListe.rowCount+" lignes");
```

List.rowHeight

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
OccurrenceListe.rowHeight
```

Description

Propriété : hauteur, en pixels, de chaque ligne dans la liste. Les paramètres de police n'agrandissent pas les lignes pour qu'elles s'adaptent en conséquence. La définition de la propriété `rowHeight` est donc la meilleure façon de s'assurer que les éléments sont affichés dans leur intégralité. La valeur par défaut est 20.

Exemple

L'exemple suivant définit les lignes à 30 pixels :

```
maListe.rowHeight = 30;
```

List.scroll

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(scroll){
  // votre code ici
}
```

Usage 2 :

```
listenerObject = new Object();
objetDécoute.scroll = function(ObjetEvnt){
  // votre code ici
}
OccurrenceListe.addEventListener("scroll", ObjetDécoute)
```

Objet événement

Outre les propriétés standard de l'objet événement, l'événement `scroll` possède une propriété supplémentaire : `direction`. Il s'agit d'une chaîne ayant deux valeurs possibles : `"horizontal"` ou `"vertical"`. Pour un événement scroll ComboBox, la valeur est toujours `"vertical"`.

Description

Evénement : diffuse à tous les écouteurs enregistrés lorsqu'une liste défile.

Le premier exemple d'utilisation utilise un gestionnaire `on()` et doit être directement lié à une occurrence de composant List. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de comportement. Par exemple, le code suivant, lié à l'occurrence de composant List `maListe`, envoie « _level0.maListe » au panneau Sortie :

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`OccurrenceListe`*) distribue un événement (ici, `scroll`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

L'exemple suivant envoie le nom de l'occurrence du composant ayant généré l'événement `change` au panneau de sortie :

```
form.scroll = function(objEvnt){
  trace("liste défilée");
}
maListe.addEventListener("scroll", form);
```

List.selectable

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`OccurrenceListe``.selectable`

Description

Propriété : valeur booléenne indiquant si la liste est sélectionnable (`true`) ou non (`false`). La valeur par défaut est `true`.

List.selectedIndex

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceListe.selectedIndex

Description

Propriété : index sélectionné d'une liste à sélection unique. La valeur est undefined si rien n'est sélectionné. Elle est égale au dernier élément sélectionné s'il y a plusieurs sélections. Si vous affectez une valeur à `selectedIndex`, toute sélection courante est effacée et l'élément indiqué est sélectionné.

Exemple

Cet exemple sélectionne l'élément après l'élément actuellement sélectionné. Si rien n'est sélectionné, l'élément 0 est sélectionné, comme suit :

```
var selIndex = maListe.selectedIndex;
maListe.selectedIndex = (selIndex==undefined ? 0 : selIndex+1);
```

Consultez également

`List.selectedIndices`, `List.selectedItem`, `List.selectedItems`

List.selectedIndices

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceListe.selectedIndices

Description

Propriété : tableau des indices des éléments sélectionnés. L'affectation de cette propriété remplace la sélection courante. La définition de `selectedIndices` à un tableau de longueur 0 (ou undefined) efface la sélection courante. La valeur est undefined si rien n'est sélectionné.

La propriété `selectedIndices` est listée dans l'ordre de sélection des éléments. Si vous cliquez successivement sur les deuxième, troisième et premier éléments, `selectedIndices` renvoie `[1,2,0]`.

Exemple

L'exemple suivant obtient les indices sélectionnés :

```
var selIndices = maListe.selectedIndices;
```

L'exemple suivant sélectionne quatre éléments :

```
var monTableau = new Array (1,4,5,7);
maListe.selectedIndices = monTableau ;
```

Consultez également

`List.selectedIndex`, `List.selectedItem`, `List.selectedItems`

List.selectedItem

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`OccurrenceListe``.selectedItem`

Description

Propriété (lecture seule) : objet d'élément dans une liste à sélection unique. (Dans une liste à sélection multiple où plusieurs éléments sont sélectionnés, `selectedItem` renvoie l'élément qui a été sélectionné le plus récemment). S'il n'y a aucune sélection, la valeur est undefined.

Exemple

Cet exemple affiche l'étiquette sélectionnée :

```
trace(maListe.selectedItem.label);
```

Consultez également

`List.selectedIndex`, `List.selectedIndices`, `List.selectedItems`

List.selectedItems

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`OccurrenceListe``.selectedItems`

Description

Propriété (lecture seule) : tableau des objets d'élément sélectionnés. Dans une liste à sélection multiple, `selectedItems` vous permet d'accéder au jeu d'éléments sélectionnés comme objets d'éléments.

Exemple

L'exemple suivant obtient un tableau des objets d'éléments sélectionnés :

```
var monTabObj = maListe.selectedItems;
```

Consultez également

`List.selectedIndex`, `List.selectedItem`, `List.selectedIndices`

List.setPropertiesAt()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`OccurrenceListe``.setPropertiesAt(`*`index`*`, `*`styleObj`*`)`

Paramètres

`index` Nombre supérieur à zéro ou inférieur à `List.length` indiquant l'index de l'élément à modifier.

`styleObj` Objet énumérant les propriétés et valeurs à définir.

Renvoie

Rien.

Description

Méthode : applique les propriétés spécifiées par le paramètre *`styleObj`* à l'élément spécifié par le paramètre *`index`*. Les propriétés supportées sont `icon` et `backgroundColor`.

Exemple

L'exemple suivant change le quatrième élément en noir et lui attribue une icône :

```
maListe.setPropertiesAt(3, {backgroundColor:0x000000, icon: "file"});
```

List.sortItems()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`OccurrenceListe``.sortItems(`*`compareFunc`*`)`

Paramètres

`compareFunc` Référence à une fonction. Cette fonction est utilisée pour comparer deux éléments afin de déterminer leur ordre de tri.

Pour plus d'informations, consultez Pour plus d'informations, consultez `Array.sort()` dans le Dictionnaire ActionScript de l'aide.

Renvoie

L'index auquel l'élément est ajouté.

Description

Méthode : trie les éléments de la liste en fonction du paramètre *compareFunc*.

Exemple

Les exemples suivants trient les éléments en fonction de leurs étiquettes en majuscules. Notez que les paramètres `a` et `b` transmis à la fonction sont des éléments qui ont des propriétés `label` et `data` :

```
maListe.sortItems(upperCaseFunc);
function upperCaseFunc(a,b){
    return a.label.toUpperCase() > b.label.toUpperCase();
}
```

Consultez également

`List.sortItemsBy()`

List.sortItemsBy()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`OccurrenceListe``.sortItemsBy(`*`NomDeChamp, ordre`*`)`

Paramètres

`NomDeChamp` Chaîne spécifiant le nom de la propriété devant être utilisée pour le tri. Cette valeur est généralement `"label"` ou `"data"`.

`ordre` Chaîne spécifiant si le tri des éléments doit être effectué par ordre croissant (`"ASC"`) ou décroissant (`"DESC"`).

Renvoie

Rien.

Description

Méthode : trie les éléments de la liste par ordre alphabétique ou numérique, dans l'ordre spécifié, avec le *`nomDeChamp`* spécifié. Si les éléments *`nomDeChamp`* sont une combinaison de chaînes de texte et d'entiers, ce sont les éléments entiers qui sont indiqués en premier. Le paramètre *`nomDeChamp`* est généralement `label` ou `data`, mais vous pouvez spécifier n'importe quelle primitive.

Exemple

Le code suivant trie les éléments de la liste `menuDeNoms` par ordre croissant en utilisant leurs étiquettes.

```
menuDeNoms.sortItemsBy("label", "ASC");
```

Consultez également

`List.sortItems()`

List.vPosition

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`OccurrenceListe``.vPosition`

Description

Propriété . fait défiler la liste pour que index soit l'élément le plus visible. Si l'objet sort des limites, se rend à l'index le plus proche dans les limites. La valeur par défaut est 0.

Exemple

L'exemple suivant définit la position de la liste au premier élément d'index :

```
maListe.vPosition = 0;
```

List.vScrollPolicy

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`OccurrenceListe``.vScrollPolicy`

Description

Propriété : chaîne déterminant si la liste supporte ou non le défilement vertical. Cette propriété peut être l'une des valeurs suivantes : `"on"`, `"off"` ou `"auto"`. La valeur `"auto"` fait apparaître une barre de défilement lorsque cela est nécessaire.

Exemple

L'exemple suivant désactive la barre de défilement :

```
maListe.vScrollPolicy = "off";
```

Vous pouvez toujours créer un défilement en utilisant `List.vPosition`.

Consultez également

`List.vPosition`

Composant Loader

Le composant Loader est un conteneur pouvant afficher un fichier SWF ou JPEG. Vous pouvez redimensionner le contenu du chargeur, ou redimensionner le chargeur lui-même pour l'adapter à la taille du contenu. Par défaut, le contenu est dimensionné pour s'ajuster au chargeur. Vous pouvez également charger du contenu à l'exécution et contrôler la progression du chargement.

Un composant Loader ne peut recevoir le focus. Cependant, le contenu chargé dans le composant Loader peut accepter le focus et avoir ses propres interactions de focus. Pour plus d'informations sur le contrôle du focus, consultez *Création de la navigation personnalisée du focus*, page 25 ou *Classe FocusManager*, page 103.

Un aperçu en direct de chaque occurrence de Loader reflète les modifications effectuées sur les paramètres dans l'inspecteur des propriétés ou le panneau de l'Inspecteur de composants lors de la programmation.

Le contenu chargé dans un composant Loader peut être activé pour l'accessibilité. Si tel est le cas, vous pouvez utiliser le panneau Accessibilité pour le rendre accessible aux lecteurs d'écrans. Pour plus d'informations, consultez « Création de contenu accessible », dans le guide Utilisation de Flash de l'aide. Il vous faudra parfois mettre votre système d'aide à jour pour obtenir ces informations.

Utilisation du composant Loader

Vous pouvez utiliser un chargeur chaque fois que vous devez récupérer du contenu depuis un emplacement distant et le placer dans une application Flash. Par exemple, vous pouvez utiliser un chargeur pour ajouter un logo d'entreprise (fichier JPEG) dans un formulaire. Vous pouvez également utiliser un chargeur pour exploiter un travail Flash qui a déjà été terminé. Par exemple, si vous avez déjà construit une application Flash et que vous souhaitez l'étendre, vous pouvez utiliser le chargeur pour placer l'ancienne application dans une nouvelle application, éventuellement comme section d'une interface d'onglets. Dans un autre exemple, vous pouvez utiliser le composant loader dans une application qui affiche des photos. Utilisez `Loader.load()`, `Loader.percentLoaded` et `Loader.complete` pour contrôler la synchronisation des chargements d'images et afficher des barres de progression pour l'utilisateur lors du chargement.

Paramètres du composant Loader

Les paramètres suivants sont des paramètres de programmation que vous définissez pour chaque occurrence de composant Loader dans l'inspecteur des propriétés ou le panneau Inspecteur des composants :

autoload indique si le contenu doit être chargé automatiquement (true) ou s'il faut attendre jusqu'à ce que la méthode `Loader.load()` soit appelée (false). La valeur par défaut est true.

content URL absolue ou relative indiquant le fichier à charger dans le chargeur. Un chemin relatif doit être relatif au fichier SWF chargeant le contenu. L'URL doit se trouver dans le même sous-domaine que l'URL où se trouve le contenu Flash. Pour une utilisation dans le lecteur Flash Player autonome ou pour un test en mode test d'animation, tous les fichiers SWF doivent être stockés dans un même dossier et les noms de fichiers ne doivent pas inclure de spécifications de dossier ni de disque. La valeur par défaut est undefined jusqu'à ce que le chargement commence.

scaleContent indique si le contenu est redimensionné pour s'ajuster au Loader (true) ou si le Loader est redimensionné pour s'ajuster au contenu (false). La valeur par défaut est true.

ActionScript vous permet de définir des options supplémentaires pour les occurrences de Loader en utilisant ses méthodes, propriétés et événements. Pour plus d'informations, consultez Classe Loader.

Création d'une application avec le composant Loader

La procédure suivante explique comment ajouter un composant Loader à une application en cours de programmation. Dans cet exemple, le chargeur charge un logo au format JPEG depuis une URL imaginaire.

Pour créer une application avec le composant Loader, effectuez les opérations suivantes :

1. Faites glisser un composant Loader du panneau Composants jusqu'à la scène.
2. Sélectionnez le chargeur sur la scène et utilisez l'outil Transformation libre pour le dimensionner en fonction de la taille du logo d'entreprise.
3. Dans l'inspecteur des propriétés, entrez **logo** comme nom d'occurrence.
4. Sélectionnez le chargeur sur la scène et dans le panneau Inspecteur des composants et effectuez les opérations suivantes :
   - Entrez **http://corp.com/websites/logo/corplogo.jpg** pour le paramètre contentPath.

Personnalisation du composant Loader

Vous pouvez orienter un composant Loader de façon horizontale et verticale en cours de programmation et à l'exécution. Lors de la création, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez la méthode `setSize()` (consultez `UIObject.setSize()`).

Le comportement de dimensionnement du composant Loader est contrôlé par la propriété `scaleContent`. Lorsque `scaleContent = true`, le contenu est dimensionné afin de rester dans les limites du chargeur (et est redimensionné lorsque `UIObject.setSize()` est appelée). Lorsque la propriété est `scaleContent = false`, la taille du composant est fixée par rapport à celle du contenu et la méthode `UIObject.setSize()` n'a aucun effet.

Utilisation de styles avec le composant Loader

Le composant Loader n'utilise pas de styles.

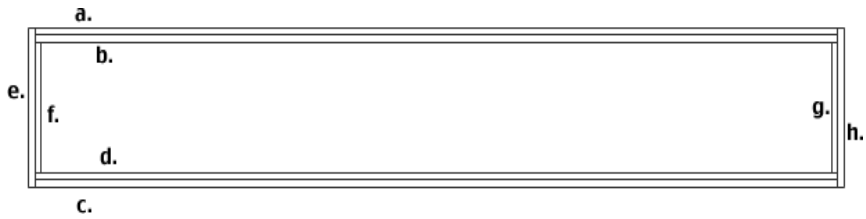
Utilisation d'enveloppes avec le composant Loader

Le composant Loader utilise RectBorder qui utilise l'API de dessin d'ActionScript. La méthode `setStyle()` (consultez `UIObject.setStyle()`) vous permet de modifier les propriétés suivantes du style RectBorder :

Styles RectBorder
`borderColor`
`highlightColor`
`borderColor`
`shadowColor`

Styles RectBorder
borderCapColor
shadowCapColor
shadowCapColor
borderCapColor

Les propriétés de style définissent les positions suivantes sur la bordure :



Classe Loader

Héritage UIObject > UIComponent > View > Loader

Espace de nom de classe ActionScript mx.controls.Loader

Les propriétés de la classe Loader vous permettent de définir le contenu de sorte qu'il se charge et contrôle sa propre progression de chargement à l'exécution.

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

Pour plus d'informations, consultez *Création de la navigation personnalisée du focus*, page 25.

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

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

Remarque : Le code suivant renvoie la valeur undefined : `trace(monOccurrencedeLoader.version);`.

Méthodes de la classe Loader

Méthode	Description
`Loader.load()`	Charge le contenu spécifié par la propriété contentPath.

Hérite de toutes les méthodes de la Classe UIObject et de la Classe UIComponent.

Propriétés de la classe Loader

Propriété	Description
`Loader.autoLoad`	Valeur booléenne indiquant si le contenu se charge automatiquement (`true`) ou si vous devez appeler `Loader.load() (false`).
`Loader.bytesLoaded`	Propriété en lecture seule indiquant le nombre d'octets ayant été chargés.
`Loader.bytesTotal`	Propriété en lecture seule indiquant le nombres d'octets du contenu.
`Loader.content`	Référence au contenu spécifié par la propriété `Loader.contentPath`. Cette propriété est en lecture seule.
`Loader.contentPath`	Chaîne indiquant l'URL du contenu devant être chargé.
`Loader.percentLoaded`	Nombre indiquant le pourcentage de contenu chargé. Cette propriété est en lecture seule.
`Loader.scaleContent`	Valeur booléenne indiquant si le contenu est dimensionné pour s'ajuster au Loader (`true`) ou si le Loader est dimensionné pour s'ajuster au contenu (`false`).

Hérite de toutes les propriétés de Classe UIObject et Classe UIComponent.

Evénements de la classe Loader

Evénement	Description
`Loader.complete`	Déclenché à la fin du chargement du contenu.
`Loader.progress`	Déclenché pendant le chargement du contenu.

Hérite de toutes les propriétés de la Classe UIObject et de la Classe UIComponent

Loader.autoLoad

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrencedeLoader`.autoLoad`

Description

Propriété : valeur booléenne indiquant s'il faut charger le contenu automatiquement (`true`), ou attendre jusqu'à ce que `Loader.load()` soit appelée (`false`). La valeur par défaut est `true`.

Exemple

Le code suivant configure le composant loader pour qu'il attende un appel `Loader.load()` :

```
loader.autoload = false;
```

Loader.bytesLoaded

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrencedeLoader.bytesLoaded

Description

Propriété (lecture seule) : nombre d'octets de contenu ayant été chargés. La valeur par défaut est 0 jusqu'à ce que le chargement commence.

Exemple

Le code suivant crée une barre de progrès et un composant Loader. Il crée ensuite un objet d'écoute avec un gestionnaire d'événement `progress` qui affiche la progression du chargement. L'écouteur est enregistré avec l'occurrence `loader` de la manière suivante :

```
createClassObject(mx.controls.ProgressBar, "pBar", 0);
createClassObject(mx.controls.Loader, "loader", 1);
loadListener = new Object();
loadListener.progress = function(objEvnt){
  // objEvnt.target est le composant ayant généré l'événement change,
  // c-à-d, le Loader.
  pBar.setProgress(loader.bytesLoaded, loader.bytesTotal); // afficher la
  progression
}
loader.addEventListener("progress", loadListener);
loader.content = "logo.swf";
```

Lorsque vous créez une occurrence avec la méthode `createClassObject()`, vous devez la placer sur la scène à l'aide des méthodes `move()` et `setSize()`. Consultez `UIObject.move()` et `UIObject.setSize()`.

Consultez également

`Loader.bytesTotal`, `UIObject.createClassObject()`

Loader.bytesTotal

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrencedeLoader.bytesTotal

Description

Propriété : (lecture seule), taille du contenu en octets. La valeur par défaut est 0 jusqu'à ce que le chargement commence.

Exemple

Le code suivant crée une barre de progrès et un composant Loader. Il crée ensuite un objet d'écoute de chargement avec un gestionnaire d'événement progress qui affiche la progression du chargement. L'écouteur est enregistré avec l'occurrence loader de la manière suivante :

```
createClassObject(mx.controls.ProgressBar, "pBar", 0);
createClassObject(mx.controls.Loader, "loader", 1);
loadListener = new Object();
loadListener.progress = function(objEvnt){
  // objEvnt.target est le composant ayant généré l'événement change,
  // c-à-d, le Loader.
  pBar.setProgress(loader.bytesLoaded, loader.bytesTotal); // afficher la
  progression
}
loader.addEventListener("progress", loadListener);
loader.content = "logo.swf";
```

Consultez également

`Loader.bytesLoaded`

Loader.complete

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(complete){
  ...
}
```

Usage 2 :

```
ObjetDécoute = new Object();
ObjetDécoute.complete = function(objetEvnt){
  ...
}
OccurrencedeLoader.addEventListener("complete", ObjetDécoute)
```

Description

Evénement : diffuse à tous les écouteurs enregistrés une fois le contenu chargé.

Le premier exemple d'utilisation utilise un gestionnaire `on()` et doit être directement associé à une occurrence de composant Loader. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de comportement. Par exemple, le code suivant, associé à l'occurrence de composant Loader `monComposantLoader`, envoie « _level0.monComposantLoader » au panneau de sortie.

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*OccurrenceListe*) distribue un événement (ici, `complete`) qui est géré par une fonction associée à un objet d'écoute (*objetDécoute*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*objetEvnt*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

L'exemple suivant crée un composant Loader puis définit un objet d'écoute avec un gestionnaire d'événement `complete` qui définit la propriété `visible` du chargeur sur `true` :

```
createClassObject(mx.controls.Loader, "loader", 0);
loadListener = new Object();
loadListener.complete = function(objEvnt){
  loader.visible = true;
}
loader.addEventListener("complete", loadListener);S
loader.contentPath = "logo.swf";
```

Loader.content

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrencedeLoader.content

Description

Propriété (lecture seule) : référence au contenu du chargeur. La valeur est undefined jusqu'à ce que le chargement commence.

Consultez également

`Loader.contentPath`

Loader.contentPath

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrencedeLoader.contentPath

Description

Propriété : chaîne indiquant l'URL absolue ou relative du fichier à charger dans le chargeur. Un chemin relatif doit être relatif au fichier SWF chargeant le contenu. L'URL doit se trouver dans le même sous-domaine que l'URL du fichier SWF en chargement.

Pour une utilisation dans Flash Player ou pour un test en mode test d'animation, tous les fichiers SWF doivent être stockés dans le même dossier et les noms de fichiers ne doivent pas inclure d'informations de dossier ni de disque.

Exemple

L'exemple suivant demande à l'occurrence de chargeur d'afficher le contenu du fichier logo.swf :

```
loader.contentPath = "logo.swf";
```

Loader.load()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrencedeLoader.load(*path*)

Paramètres

path Paramètre facultatif spécifiant la valeur de la propriété `contentPath` avant le début du chargement. Si aucune valeur n'est spécifiée, la valeur courante de `contentPath` est utilisée telle quelle.

Renvoie

Rien.

Description

Méthode : dit au chargeur de commencer le chargement de son contenu.

Exemple

Le code suivant crée une occurrence Loader et définit la propriété `autoload` sur `false` de sorte que le chargeur doit attendre un appel de la méthode `load()` pour commencer le chargement du contenu. Il appelle ensuite `load()` et indique le contenu à charger :

```
createClassObject(mx.controls.Loader, "loader", 0);
loader.autoload = false;
loader.load("logo.swf");
```

Loader.percentLoaded

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
OccurrencedeLoader.percentLoaded
```

Description

Propriété (lecture seule) : nombre indiquant le pourcentage de contenu ayant été chargé. Cette propriété est généralement utilisée pour présenter la progression du chargement à l'utilisateur de façon plus claire. Utilisez le code suivant pour arrondir le chiffre à l'entier le plus proche :

```
Math.round(bytesLoaded/bytesTotal*100))
```

Exemple

L'exemple suivant crée une occurrence Loader puis un objet d'écoute avec un gestionnaire de progression qui recherche le pourcentage chargé et l'envoie au panneau de sortie :

```
createClassObject(Loader, "loader", 0);
loadListener = new Object();
loadListener.progress = function(objEvnt){
  // objEvnt.target est le composant ayant généré l'événement change,
  // c'est-à-dire, le Loader.
  trace("logo.swf est chargé à" + loader.percentLoaded + " %."); // progression
   du chargement
}
loader.addEventListener("complete", loadListener);
loader.content = "logo.swf";
```

Loader.progress

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(progress){
   ...
}
```

Usage 2 :

```
ObjetDécoute = new Object();
objetDécoute.progress = function(objetEvnt){
   ...
}
OccurrenceDeLoader.addEventListener("progress", objetDécoute)
```

Description

Evénement : diffuse à l'ensemble des écouteurs enregistrés pendant le chargement du contenu. Cet événement est déclenché lorsque le chargement est effectué grâce au paramètre autoload ou par un appel à `Loader.load()`. L'événement progress n'est pas toujours diffusé. L'événement `complete` peut être diffusé sans qu'aucun événement `progress` ne soit distribué. Ceci peut particulièrement se produire si le contenu chargé est un fichier local.

Le premier exemple d'utilisation utilise un gestionnaire `on()` et doit être directement associé à une occurrence de composant Loader. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de comportement. Par exemple, le code suivant, associé à l'occurrence de composant Loader `monComposantLoader`, envoie « _level0.monComposantLoader » au panneau de sortie.

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`OccurrenceDeLoader`*) distribue un événement (ici, `progress`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

Le code suivant crée une occurrence de Loader, puis un objet d'écoute avec un gestionnaire d'événement pour l'événement progress. Celui-ci envoie un message au panneau de sortie concernant le pourcentage du contenu qui a été chargé :

```
createClassObject(mx.controls.Loader, "loader", 0);
loadListener = new Object();
loadListener.progress = function(objEvnt){
  // objEvnt.target est le composant ayant généré l'événement change,
  // c'est-à-dire, le Loader.
  trace("logo.swf est chargé à" + loader.percentLoaded + " %."); // progression
  du chargement
}
loader.addEventListener("progress", loadListener);
loader.contentPath = "logo.swf";
```

Loader.scaleContent

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
OccurrenceDeLoader.scaleContent
```

Description

Propriété : indique si la taille du contenu s'ajuste pour correspondre au Loader (true), ou si le Loader s'ajuste pour correspondre au contenu (false). La valeur par défaut est true.

Exemple

Le code suivant demande au Loader de modifier sa taille en fonction de la taille de son contenu :

```
loader.strechContent = false;
```

Composant MediaController

Pour des informations actualisées sur cette fonction, cliquez sur le bouton Mettre à jour dans la partie supérieure de l'onglet Aide.

Composant MediaDisplay

Pour des informations actualisées sur cette fonction, cliquez sur le bouton Mettre à jour dans la partie supérieure de l'onglet Aide.

Composant MediaPlayback

Pour des informations actualisées sur cette fonction, cliquez sur le bouton Mettre à jour dans la partie supérieure de l'onglet Aide.

Composant Menu

Pour des informations actualisées sur cette fonction, cliquez sur le bouton Mettre à jour dans la partie supérieure de l'onglet Aide.

Composant MenuBar

Pour des informations actualisées sur cette fonction, cliquez sur le bouton Mettre à jour dans la partie supérieure de l'onglet Aide.

Composant NumericStepper

Le composant NumericStepper permet à un utilisateur de faire défiler un ensemble ordonné de nombres. Il s'agit d'un nombre affiché à côté de petits boutons fléchés vers le haut et vers le bas. Lorsqu'un utilisateur appuie sur les flèches, le nombre augmente ou diminue de façon incrémentielle. Si l'utilisateur clique sur l'un des boutons fléchés, le nombre augmente ou diminue, en fonction de la valeur du paramètre stepSize, jusqu'à ce que l'utilisateur relâche le bouton de la souris ou que la valeur minimale ou maximale soit atteinte.

Le composant NumericStepper gère uniquement les données numériques. Vous devez également redimensionner le stepper lors de la programmation pour afficher plus de deux chiffres (par exemple, les nombres 5246 ou 1.34).

Un stepper peut être activé ou désactivé dans une application. Lorsqu'il est désactivé, le stepper ne reçoit aucune information provenant du clavier ou de la souris. Un stepper activé reçoit le focus si vous cliquez dessus ou si vous utilisez la tabulation pour l'ouvrir et son focus interne est défini dans le champ de texte. Lorsqu'une occurrence NumericStepper a le focus, vous pouvez utiliser les touches suivantes pour la contrôler :

Touche	Description
Bas	Les valeurs sont modifiées d'une unité.
Gauche	Déplace le point d'insertion vers la gauche à l'intérieur du champ de texte.
Droite	Déplace le point d'insertion vers la droite à l'intérieur du champ de texte.
Maj +Tab	Place le focus sur l'objet précédent.
Tab	Place le focus sur l'objet suivant.
Haut	Les valeurs sont modifiées d'une unité.

Pour plus d'informations sur le contrôle du focus, consultez *Création de la navigation personnalisée du focus*, page 25 ou *Classe FocusManager*, page 103.

Un aperçu en direct de chaque occurrence de stepper reflète la valeur du paramètre value indiqué par le panneau de l'inspecteur des propriétés ou des composants au cours de la programmation. Cependant, il n'y a aucune interaction entre le clavier ou la souris et les boutons du stepper dans l'aperçu en direct.

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

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

Vous n'activez l'accessibilité d'un composant qu'une fois, quel que soit son nombre d'occurrences. Pour plus d'informations, consultez « Création de contenu accessible », dans le guide Utilisation de Flash de l'aide. Il vous faudra parfois mettre votre système d'aide à jour pour obtenir ces informations.

Utilisation du composant NumericStepper

Le NumericStepper peut être utilisé là où vous souhaitez qu'un utilisateur sélectionne une valeur numérique. Par exemple, vous pouvez utiliser un composant NumericStepper dans un formulaire pour permettre à un utilisateur de définir la date d'expiration de sa carte de crédit. Dans un autre exemple, vous pouvez utiliser un NumericStepper pour permettre à un utilisateur d'augmenter ou de diminuer la taille d'une police.

Paramètres de NumericStepper

Les paramètres suivants sont des paramètres de programmation que vous pouvez définir pour chaque occurrence de composant NumericStepper dans le panneau de l'inspecteur des propriétés ou des composants :

value définit la valeur de l'incrément actuel. La valeur par défaut est 0.

minimum définit la valeur minimum. La valeur par défaut est 0.

maximum définit la valeur maximum. La valeur par défaut est 10.

stepSize définit l'unité d'incrémentation de la valeur. La valeur par défaut est 1.

Vous pouvez rédiger du code ActionScript pour contrôler ces options et d'autres options des composants NumericStepper à l'aide des propriétés, méthodes et événements ActionScript. Pour plus d'informations, consultez Classe NumericStepper.

Création d'une application avec le composant NumericStepper

La procédure suivante explique comment ajouter un composant NumericStepper à une application lors de la programmation. Dans cet exemple, le stepper permet à l'utilisateur de choisir une classification d'animation comprise entre 0 et 5 étoiles par incréments d'une demie étoile.

Pour créer une application avec le composant Button, procédez comme suit :

1 Faites glisser un composant NumericStepper du panneau Composants jusqu'à la scène.

2 Dans l'inspecteur des propriétés, entrez **stepperEtoile** comme nom d'occurrence.

3 Dans l'inspecteur des propriétés, procédez comme suit :

- Entrez 0 pour le paramètre minimum.
- Entrez 5 pour le paramètre maximum.
- Entrez .5 pour le paramètre stepSize.
- Entrez 0 pour le paramètre value.

4 Choisissez l'image 1 dans le scénario, ouvrez le panneau Actions et saisissez le code suivant :

```
movieRate = new Object();
movieRate.change = function (objetEvnt){
  classementEtoiles.value = objetEvnt.target.value;
}
stepperEtoile.addEventListener("change", movieRate);
```

La dernière ligne de code ajoute un gestionnaire d'événement `change` à l'occurrence `stepperEtoile`. Le gestionnaire définit le clip `classementEtoiles` pour afficher la quantité d'étoiles indiquée dans l'occurrence `stepperEtoile`. (Pour voir ce code fonctionner, vous devez créer un clip `classementEtoiles` avec une propriété `value` qui affiche les étoiles.)

Personnalisation du composant NumericStepper

Vous pouvez orienter un composant NumericStepper horizontalement et verticalement à la fois en cours de programmation et à l'exécution. Lors de la création, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez la méthode `setSize()` (consultez `UIObject.setSize()`) ou toute propriété et méthode applicable de la classe NumericStepper. Pour plus d'informations, consultez Classe NumericStepper.

Le redimensionnement du composant NumericStepper ne modifie pas la taille des boutons fléchés vers le haut et le bas. Si le stepper est redimensionné au-delà de la hauteur par défaut, les boutons du stepper sont placés en haut et en bas du composant. Les boutons du stepper apparaissent toujours à droite du champ de texte.

Utilisation de styles avec le composant NumericStepper

Vous pouvez définir des propriétés de style pour modifier l'apparence d'une occurrence stepper. Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, consultez *Utilisation des styles pour personnaliser la couleur et le texte des composants*, page 27.

Un composant NumericStepper supporte les styles de halo suivants :

Style	Description
`themeColor`	Arrière-plan d'un composant. Il s'agit du seul style de couleur qui n'hérite pas de sa valeur. "haloGreen", "haloBlue" et "haloOrange" sont des valeurs possibles.
`color`	Texte d'une étiquette de composant.
`disabledColor`	Couleur désactivée pour du texte.
`fontFamily`	Nom de police pour du texte.
`fontSize`	Taille en points pour la police.
`fontStyle`	Style de la police : "normal" ou "italic".
`fontWeight`	Epaisseur de la police : "normal" ou "bold".
`textDecoration`	Décoration du texte : "none" ou "underline".
`textAlign`	Alignement du texte : "left", "right" ou "center".

Utilisation d'enveloppes avec le composant NumericStepper

Le composant NumericStepper utilise des enveloppes pour représenter ses états visuels. Pour envelopper le composant NumericStepper lors de la programmation, modifiez les symboles d'enveloppe dans la bibliothèque et exportez de nouveau le composant en tant que fichier SWC. Les symboles d'enveloppe sont situés dans le dossier Flash UI Components 2/Themes/MMDefault/Stepper Elements/states de la bibliothèque. Pour plus d'informations, consultez *A propos de l'application des enveloppes aux composants*, page 37.

Si un stepper est activé, les boutons fléchés vers le haut et vers le bas affichent leur état survolé lorsque le pointeur se déplace au-dessus d'eux. Les boutons affichent leur état enfoncé lorsque l'utilisateur clique dessus. Les boutons reviennent à l'état survolé lorsque le bouton de la souris est relâché. Si le pointeur s'éloigne des boutons alors que le bouton de la souris est enfoncé, les boutons reviennent à leur état original.

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

Un composant NumericStepper utilise les propriétés d'enveloppe suivantes :

Propriété	Description
`upArrowUp`	L'état haut de la flèche vers le haut. La valeur par défaut est `StepUpArrowUp`.
`upArrowDown`	Etat enfoncé de la flèche vers le haut. La valeur par défaut est `StepUpArrowDown`.
`upArrowOver`	Etat Survolé de la flèche vers le haut. La valeur par défaut est `StepUpArrowOver`.
`upArrowDisabled`	Etat désactivé de la flèche vers le haut. La valeur par défaut est `StepUpArrowDisabled`.
`downArrowUp`	Etat Relevé de la flèche bas. La valeur par défaut est `StepDownArrowUp`.
`downArrowDown`	Etat Enfoncé de la flèche bas. La valeur par défaut est `StepDownArrowDown`.
`downArrowOver`	Etat Survolé de la flèche bas. La valeur par défaut est `StepDownArrowOver`.
`downArrowDisabled`	Etat désactivé de la flèche bas. La valeur par défaut est `StepDownArrowDisabled`.

Classe NumericStepper

Héritage UIObject > UIComponent > NumericStepper

Nom de classe ActionScript mx.controls.NumericStepper

Les propriétés de la classe NumericStepper vous permettent d'ajouter et d'indiquer les valeurs minimum et maximum, l'unité d'incrémentation et la valeur courante à l'exécution.

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

Le composant NumericStepper utilise le FocusManager pour annuler le rectangle de focus de Flash Player et dessiner un rectangle de focus personnalisé avec des angles arrondis. Pour plus d'informations, consultez *Création de la navigation personnalisée du focus*, page 25.

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

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

Remarque : Le code suivant renvoie la valeur undefined : `trace(monOccurrenceDeNumericStepper.version);`.

Méthodes de la classe NumericStepper

Hérite de toutes les propriétés de la Classe UIObject et de la Classe UIComponent.

Propriétés de la classe NumericStepper

Propriété	Description
NumericStepper.maximum	Nombre indiquant la valeur de plage maximum.
NumericStepper.minimum	Nombre indiquant la valeur de plage minimum.
NumericStepper.nextValue	Nombre indiquant la prochaine valeur séquentielle. Cette propriété est en lecture seule.
NumericStepper.previousValue	Nombre indiquant la valeur séquentielle précédente. Cette propriété est en lecture seule.
NumericStepper.stepSize	Nombre indiquant l'unité d'incrémentation.
NumericStepper.value	Nombre indiquant la valeur courante du stepper.

Hérite de toutes les propriétés de Classe UIObject et Classe UIComponent.

Evénements de la classe NumericStepper

Evénement	Description
NumericStepper.change	Déclenché lorsque la valeur change.

Hérite de toutes les propriétés de Classe UIObject et Classe UIComponent.

NumericStepper.change

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

```
ObjetDécoute = new Object();
objetDécoute.change = function(objetEvnt){
   ...
}
OccurrenceDeStepper.addEventListener("change", objetDécoute)
```

Description

Evénement : diffuse à l'ensemble des écouteurs enregistrés lorsque la valeur du stepper est modifiée.

Le premier exemple d'utilisation utilise un gestionnaire `on()` et doit être directement associé à une occurrence de composant NumericStepper. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de comportement. Par exemple, le code suivant, associé à l'incrémenteur `monStepper`, envoie « _level0.monStepper » au panneau de sortie.

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`OccurrenceDIncrémenteur`*) distribue un événement (ici, `change`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

Cet exemple, écrit sur une image du scénario, envoie un message au panneau de sortie lorsqu'un incrémenteur nommé `monIncrémenteurNumérique` est modifié. La première ligne de code crée un objet d'écoute intitulé `form`. La deuxième ligne définit une fonction pour l'événement `change` sur l'objet d'écoute. La fonction comporte une action `trace` qui utilise l'objet événement automatiquement transmis à cette fonction, ici `objEvnt`, pour générer un message. La propriété `target` d'un objet d'écoute est le composant ayant généré l'événement, dans cet exemple, `monIncrémenteurNumérique`. L'utilisateur accède à la propriété `NumericStepper.value` à partir de la propriété `target` de l'objet événement. La dernière ligne appelle la méthode `UIEventDispatcher.addEventListener()` depuis `monIncrémenteurNumérique` et lui transmet l'événement `change` et l'objet d'écoute `form` comme paramètres, comme dans l'exemple suivant :

```
form = new Object();
form.change = function(objEvnt){
  // objEvnt.target est le composant ayant généré l'événement change,
  //c'est-à-dire, l'incrémenteur numérique.
  trace("Valeur passée à " + objEvnt.target.value);
}
monIncrémenteurNumérique.addEventListener("change", form);
```

NumericStepper.maximum

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
OccurrenceDeStepper.maximum
```

Description

Propriété : la valeur de plage maximum de l'incrémenteur. Cette propriété peut contenir un nombre comprenant jusqu'à trois chiffres. La valeur par défaut est 10.

Exemple

L'exemple suivant définit la valeur maximum de la plage de l'incrémenteur à 20 :

```
monIncrémenteur.maximum = 20;
```

Consultez également

NumericStepper.minimum

NumericStepper.minimum

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
OccurrenceDeStepper.minimum
```

Description

Propriété : la valeur de plage minimum du stepper. Cette propriété peut contenir un nombre comprenant jusqu'à trois chiffres. La valeur par défaut est 0.

Exemple

L'exemple suivant définit la valeur minimum de la plage du stepper à 20 :

```
monIncrémenteur.minimum = 100;
```

Consultez également

NumericStepper.maximum

NumericStepper.nextValue

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeStepper.nextValue

Description

Propriété (lecture seule) : prochaine valeur séquentielle. Cette propriété peut contenir un nombre comprenant jusqu'à trois chiffres.

Exemple

L'exemple suivant définit la propriété `stepSize` à 1 et la valeur de départ à 4, ce qui amène la valeur de `nextValue` à 5 :

```
monIncrémenteur.stepSize = 1;
monIncrémenteur.value = 4;
trace(monIncrémenteur.nextValue);
```

Consultez également

NumericStepper.previousValue

NumericStepper.previousValue

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeStepper.previousValue

Description

Propriété (lecture seule) : valeur séquentielle précédente. Cette propriété peut contenir un nombre comprenant jusqu'à trois chiffres.

Exemple

L'exemple suivant définit la propriété `stepSize` à 1 et la valeur de départ à 4, ce qui amène la valeur de `nextValue` à 3 :

```
monIncrémenteur.stepSize = 1;
monIncrémenteur.value = 4;
trace(monIncrémenteur.previousValue);
```

Consultez également

NumericStepper.nextValue

NumericStepper.stepSize

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeStepper.stepSize

Description

Propriété : la quantité d'unités à modifier à partir de la valeur courante. La valeur par défaut est 1. Cette valeur ne peut être 0. Cette propriété peut contenir un nombre comprenant jusqu'à trois chiffres.

Exemple

L'exemple suivant règle la valeur de `value` et de l'unité `stepSize` sur 2. La valeur de `nextValue` est 4 :

```
monIncrémenteur.value = 2;
monIncrémenteur.stepSize = 2;
trace(monIncrémenteur.nextValue);
```

NumericStepper.value

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeStepper.value

Description

Propriété : la valeur courante est affichée dans la zone de texte du stepper. La valeur ne sera pas affectée si elle ne correspond pas à la plage du stepper et à l'unité d'incrémentation telles que définies dans la propriété `stepSize`. Cette propriété peut contenir un nombre comprenant jusqu'à trois chiffres.

Exemple

L'exemple suivant définit la `valeur` courante du stepper à 10 et envoie la valeur au panneau de sortie :

```
monIncrémenteur.value = 10;
trace(monIncrémenteur.value);
```

Classe PopUpManager

Espace de nom de classe ActionScript mx.managers.PopUpManager

La classe PopUpManager vous permet de créer des fenêtres chevauchées qui peuvent être modales ou non modales. (Une fenêtre modale ne permet pas d'interagir avec d'autres fenêtres lorsqu'elle est active.) Vous pouvez appeler `PopUpManager.createPopUp()` pour créer une fenêtre chevauchée et appeler `PopUpManager.deletePopUp()` sur l'occurrence de la fenêtre pour détruire une fenêtre contextuelle.

Méthodes de la classe PopUpManager

Evénement	Description
`PopUpManager.createPopUp()`	Crée une fenêtre contextuelle.
`PopUpManager.deletePopUp()`	Supprime une fenêtre contextuelle créée par un appel à `PopUpManager.createPopUp()`.

PopUpManager.createPopUp()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004 et Flash MX Professionnel 2004

Usage

```
PopUpManager.createPopUp(parent, classe, modale [, objinit, EvntExtérieurs])
```

Paramètres

parent Référence à une fenêtre par dessus laquelle afficher une fenêtre contextuelle.

classe Référence à la classe de l'objet à créer.

modale Valeur booléenne indiquant si la fenêtre est modale (`true`) ou non (`false`).

objinit Objet contenant les propriétés d'initialisation. Ce paramètre est facultatif.

EvntExtérieurs Valeur booléenne indiquant si un événement est déclenché lorsque l'utilisateur clique en dehors de la fenêtre (`true`) ou non (`false`). Ce paramètre est facultatif.

Renvoie

Référence à la fenêtre créée.

Description

Méthode : si elle est modale, un appel à `createPopUp()` trouve la fenêtre commençant par parent et crée une occurrence de la classe. Si elle est non modale, un appel à `createPopUp()` crée une occurrence de la classe comme enfant de la fenêtre parent.

Exemple

Le code suivant crée une fenêtre modale lorsque l'utilisateur clique sur le bouton :

```
lo = new Object();
lo.click = function(){
  mx.managers.PopUpManager.createPopUp(_root, mx.containers.Window, true);
}
button.addEventListener("click", lo);
```

PopUpManager.deletePopUp()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004 et Flash MX Professionnel 2004

Usage

```
occurrenceFenêtre.deletePopUp();
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : supprime une fenêtre contextuelle ainsi que l'état modal. C'est à la fenêtre chevauchée d'appeler `PopUpManager.deletePopUp()` lorsque la fenêtre est détruite.

Exemple

Le code suivant crée une fenêtre modale, appelée `fen` et dotée d'un bouton de fermeture, et la supprime lorsqu'un utilisateur clique sur ce bouton :

```
import mx.managers.PopUpManager
import mx.containers.Window
fen = PopUpManager.createPopUp(_root, Window, true, {closeButton:true});
lo = new Object();
lo.click = function(){
  fen.deletePopUp();
}
fen.addEventListener("click", lo);
```

Composant ProgressBar

Le composant ProgressBar affiche la progression du chargement lorsqu'un utilisateur attend que le contenu soit chargé. Le processus de chargement peut être déterminé ou indéterminé. Une barre de progrès déterminée est une représentation linéaire de la progression d'une tâche dans le temps. Elle est utilisée lorsque la quantité de contenu à charger est connue. Une barre de progrès indéterminée est utilisée lorsque la quantité de contenu à charger est inconnue. Vous pouvez ajouter une étiquette pour afficher la progression du contenu en chargement.

De par leur configuration, les composants sont exportés dans la première image par défaut. Ils sont donc chargés dans une application avant le rendu de la première image. Pour créer un preloader pour une application, vous devez désactiver l'option Exporter dans la première image dans la boîte de dialogue Propriétés de liaison de chaque composant (menu d'options du panneau Bibliothèque > Liaison). Vous devez toutefois configurer le composant ProgressBar de sorte qu'il soit exporté dans la première image car son affichage doit précéder la lecture du reste du contenu dans Flash Player.

Un aperçu en direct de chaque occurrence ProgressBar reflète les modifications effectuées sur les paramètres dans le panneau de l'inspecteur des propriétés ou des composants en cours de programmation. Les paramètres suivants sont reflétés dans l'aperçu en direct : conversion, direction, label, labelPlacement, mode et source.

Utilisation du composant ProgressBar

Une barre de progrès vous permet d'afficher la progression du contenu pendant le chargement. Ces informations sont essentielles pour l'utilisateur lorsqu'il interagit avec une application.

Vous pouvez utiliser le composant ProgressBar dans plusieurs modes. Définissez le mode à l'aide du paramètre mode. Les modes les plus communément utilisés sont "event" et "polled". Ces méthodes utilisent le paramètre `source` pour spécifier un processus de chargement qui émet des événements `progress` et `complete` (mode event) ou expose des méthodes `getBytesLoaded` et `getsBytesTotal` (mode polled). Vous pouvez également utiliser le composant ProgressBar en mode manuel en définissant manuellement les propriétés `maximum`, `minimum` et `indeterminate` ainsi que les appels à la méthode `ProgressBar.setProgress()` .

Paramètres du composant ProgressBar

Les paramètres suivants sont des paramètres de programmation que vous pouvez définir pour chaque occurrence de composant ProgressBar dans le panneau de l'inspecteur des propriétés ou des composants :

mode Mode dans lequel opère la barre de progrès. Cette valeur peut être : event, polled ou manual. La valeur par défaut est event.

source Chaîne devant être convertie en objet représentant le nom d'occurrence de la source.

direction La direction dans laquelle avance la barre de progrès. Cette valeur peut être right ou left, la valeur par défaut étant right.

label Texte indiquant la progression du chargement. Ce paramètre est une chaîne au format « %1 sur %2 chargé (%3%%) »; %1 est un espace réservé pour le nombre d'octets actuellement chargés, %2 est un espace réservé pour le nombre total d'octets chargés et %3 est un espace réservé pour le pourcentage de contenu chargé. Les caractères « %% » sont un espace réservé pour le caractère « % ». Si une valeur pour %2 est inconnue, elle est remplacée par « ?? ». Si une valeur est undefined, l'étiquette ne s'affiche pas.

labelPlacement Position de l'étiquette par rapport à la barre de progrès. Ce paramètre peut prendre l'une des valeurs suivantes : top, bottom, left, right, center. La valeur par défaut est bottom.

conversion Nombre pour diviser les valeurs %1 et %2 dans la chaîne de l'étiquette avant qu'elles ne soient affichées. La valeur par défaut est 1.

Vous pouvez rédiger du code ActionScript pour contrôler ces options ainsi que d'autres options des composants ProgressBar à l'aide des propriétés, méthodes et événements ActionScript. Pour plus d'informations, consultez Classe ProgressBar.

Création d'une application avec le composant ProgressBar

La procédure suivante explique comment ajouter un composant ProgressBar à une application lors de la programmation. Dans cet exemple, la barre de progrès est utilisée en mode event. En mode event, le contenu en cours de chargement doit émettre des événements `progress` et `complete` pour permettre à la barre de progrès d'afficher la progression. Le composant Loader émet ces événements. Pour plus d'informations, consultez *Composant Loader*, page 143.

Pour créer une application avec le composant ProgressBar en mode event, effectuez les opérations suivantes :

1 Faites glisser un composant ProgressBar du panneau Composants jusqu'à la scène.

2 Dans l'inspecteur des propriétés, procédez comme suit :

- Entrez **barreP** comme nom d'occurrence.
- Sélectionnez le paramètre de mode event.

3 Faites glisser un composant Loader du panneau Composants jusqu'à la scène.

4 Dans l'inspecteur des propriétés, entrez **loader** comme nom d'occurrence.

5 Sélectionnez la barre de progrès sur la scène, et, dans l'inspecteur des propriétés, entrez **loader** comme paramètre de source.

6 Sélectionnez l'image 1 dans le scénario, ouvrez le panneau Actions et entrez le code suivant qui charge un fichier JPEG dans le composant Loader :

```
loader.autoLoad = false;
loader.content = "http://imagecache2.allposters.com/images/86/
  017_PP0240.jpg";
barreP.source = loader;
// le chargement ne commence que lorsque la méthode load est appelée
loader.load();
```

Dans l'exemple suivant, la barre de progrès est utilisée en mode polled. En mode polled, la barre de progrès utilise les méthodes `getBytesLoaded` et `getBytesTotal` de l'objet source pour afficher la progression.

Pour créer une application avec le composant ProgressBar en mode polled, effectuez les opérations suivantes :

1 Faites glisser un composant ProgressBar du panneau Composants jusqu'à la scène.

2 Dans l'inspecteur des propriétés, procédez comme suit :

- Entrez **barreP** comme nom d'occurrence.
- Sélectionnez le paramètre de mode polled.
- Entrez **loader** comme paramètre de source.

3 Sélectionnez l'image 1 dans le scénario, ouvrez le panneau Actions et entrez le code suivant qui crée un objet Sound nommé `loader` et appelle la méthode `loadSound()` pour charger un son dans l'objet Sound :

```
var loader:Object = new Sound();
loader.loadSound("http://soundamerica.com/sounds/sound_fx/A-E/air.wav",
  true);
```

Dans l'exemple suivant, la barre de progrès est utilisée en mode manuel**.** En mode manuel, vous devez définir les propriétés `maximum`, `minimum` et `indeterminate` en conjonction avec la méthode `setProgress()` pour afficher la progression. Vous ne définissez pas la propriété `source` en mode manuel.

Pour créer une application avec le composant ProgressBar en mode manual, procédez comme suit :

1 Faites glisser un composant ProgressBar du panneau Composants jusqu'à la scène.

2 Dans l'inspecteur des propriétés, procédez comme suit :

- Entrez **barreP** comme nom d'occurrence.
- Sélectionnez le paramètre de mode manual.

3 Sélectionnez l'image 1 dans le scénario, ouvrez le panneau Actions et entrez le code suivant qui met à jour manuellement la barre de progrès sur chaque téléchargement de fichier en appelant la méthode `setProgress()` :

```
for(var:Number i=1; i <= total; i++){
  // insérez le code pour charger le fichier
  // insérez le code pour charger le fichier
barreP.setProgress(i,total);
}
```

Personnalisation du composant ProgressBar

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

Les embouts gauche et droit de la barre de progrès et le graphique de suivi ont une taille fixe. Lorsque vous redimensionnez une barre de progrès, sa partie centrale est redimensionnée pour s'ajuster entre les embouts. Si une barre de progrès est trop petite, il se peut que le rendu soit incorrect.

Utilisation de styles avec le composant ProgressBar

Vous pouvez définir des propriétés de style pour modifier l'apparence d'une occurrence de barre de progrès. Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, consultez *Utilisation des styles pour personnaliser la couleur et le texte des composants*, page 27.

Un composant ProgressBar supporte les styles de halo suivants :

Style	Description
`themeColor`	Arrière-plan d'un composant. Il s'agit du seul style de couleur qui n'hérite pas de sa valeur. "haloGreen", "haloBlue" et "haloOrange" sont des valeurs possibles.
`color`	Texte d'une étiquette de composant.
`disabledColor`	Couleur désactivée pour du texte.
`fontFamily`	Nom de police pour du texte.
`fontSize`	Taille en points pour la police.
`fontStyle`	Style de la police : "normal" ou "italique".
`fontWeight`	Epaisseur de la police : "normal" ou "gras".
`textDecoration`	Décoration du texte : "aucun" ou "souligné".

Utilisation d'enveloppes avec le composant ProgressBar

Le composant ProgressBar utilise les symboles de clip suivants pour afficher ses états : TrackMiddle, TrackLeftCap, TrackRightCap et BarMiddle, BarLeftCap, BarRightCap et IndBar. Le symbole IndBar est utilisé pour une barre de progrès indéterminée. Pour appliquer une enveloppe au composant ProgressBar au cours de la programmation, modifiez les symboles dans la bibliothèque et exportez de nouveau le composant en tant que fichier SWC. Les symboles sont situés dans le dossier Flash UI Components 2/Themes/MMDefault/ProgressBar Elements dans la bibliothèque du fichier HaloTheme.fla ou du fichier SampleTheme.fla. Pour plus d'informations, consultez *A propos de l'application des enveloppes aux composants*, page 37.

Si vous utilisez la méthode `UIObject.createClassObject()` pour créer une occurrence de composant ProgressBar de façon dynamique (à l'exécution), vous pouvez également lui appliquer une enveloppe de façon dynamique. Pour appliquer un composant lors de l'exécution, définissez les propriétés d'enveloppe du paramètre *`initObject`* qui est passé à la méthode `createClassObject()`. Les propriétés d'enveloppe définissent le nom des symboles à utiliser comme états de la barre de progrès.

Un composant ProgressBar utilise les propriétés d'enveloppe suivantes :

Propriété	Description
`progTrackMiddleName`	Le milieu extensible du rail d'une barre de défilement. La valeur par défaut est `ProgTrackMiddle`.
`progTrackLeftName`	L'embout gauche à taille fixe. La valeur par défaut est `ProgTrackLeft`.

Propriété	Description
progTrackRightName	L'embout droit à taille fixe. La valeur par défaut est ProgTrackRight.
progBarMiddleName	Le graphique central extensible de la barre. La valeur par défaut est ProgBarMiddle.
progBarLeftName	L'embout gauche de barre à taille fixe. La valeur par défaut est ProgBarLeft.
progBarRightName	L'embout droit de barre à taille fixe. La valeur par défaut est ProgBarRight.
progIndBarName	Le graphique de barre indéterminé. La valeur par défaut est ProgIndBar.

Classe ProgressBar

Héritage UIObject > ProgressBar

Espace de nom de classe ActionScript mx.controls.ProgressBar

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

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

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

Remarque : Le code suivant renvoie la valeur undefined : `trace(monOccurrenceDeBarreDeProgrès.version);`.

Méthodes de la classe ProgressBar

Méthode	Description
ProgressBar.setProgress()	Définit la progression de la barre en mode manuel.

Hérite de toutes les méthodes de la Classe UIObject.

Propriétés de la classe ProgressBar

Propriété	Description
ProgressBar.conversion	Nombre utilisé pour convertir la valeur courante des octets chargés et les valeurs totales des octets chargés.
ProgressBar.direction	Direction dans laquelle la barre de progrès se remplit.
ProgressBar.indeterminate	Indique que le nombre total d'octets de la source est inconnu.
ProgressBar.label	Texte accompagnant la barre de progrès.
ProgressBar.labelPlacement	Emplacement de l'étiquette par rapport à la barre de progrès.
ProgressBar.maximum	Valeur maximum de la barre de progrès en mode manuel.
ProgressBar.minimum	Valeur minimum de la barre de progrès en mode manuel.
ProgressBar.mode	Mode dans lequel la barre de progrès charge le contenu.
ProgressBar.percentComplete	Nombre indiquant le pourcentage chargé.
ProgressBar.source	Contenu à charger dont la progression est contrôlée par la barre de progrès.
ProgressBar.value	Indique le niveau de progression effectué. Cette propriété est en lecture seule.

Hérite de toutes les propriétés de la Classe UIObject.

Evénements de la classe ProgressBar

Evénement	Description
ProgressBar.complete	Déclenché une fois le téléchargement terminé.
ProgressBar.progress	Déclenché pendant le chargement du contenu en mode event ou polled.

Hérite de tous les événements de la Classe UIObject.

ProgressBar.complete

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(complete){
  ...
}
```

Usage 2 :

```
ObjetDécoute = new Object();
ObjetDécoute.complete = function(objetEvnt){
  ...
}
barreP.addEventListener("complete", objetDécoute)
```

Objet événement

Outre les propriétés d'objet événement standard, deux propriétés supplémentaires sont définies pour l'événement `ProgressBar.complete` : `current` (la valeur chargée qui est égale au total) et `total` (la valeur totale).

Description

Evénement : diffuse à l'ensemble des écouteurs enregistrés une fois la progression du chargement terminée.

Le premier exemple d'utilisation fait appel à un gestionnaire `on()` et doit être directement associé à une occurrence de composant ProgressBar. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de comportement. Par exemple, le code suivant, associé à l'occurrence `barreP`, envoie « _level0.barreP » au panneau de sortie.

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*barreP*) distribue un événement (ici, `complete`) qui est géré par une fonction associée à un objet d'écoute (*objetDécoute*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*objetEvnt*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

Cet exemple crée un objet d'écoute `form` avec une fonction de rappel `complete` qui envoie un message au panneau de sortie avec la valeur de l'occurrence `barreP` de la manière suivante :

```
form.complete = function(ObjEvnt){
  // objEvnt.target est le composant ayant généré l'événement change,
  // c.-à-d., la barre de progrès.
  trace("Valeur changée en" + ObjEvnt.target.value);
}
barreP.addEventListener("complete", form);
```

Consultez également

`UIEventDispatcher.addEventListener()`

ProgressBar.conversion

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceBarreP`.conversion`

Description

Propriété : nombre qui définit une valeur de conversion pour les valeurs entrantes. Il divise les valeurs courantes et totales, les met au plancher et affiche la valeur convertie dans la propriété `label`. La valeur par défaut est 1.

Exemple

Le code suivant affiche la valeur de progression du chargement en kilooctets :

```
barreP.conversion = 1024;
```

ProgressBar.direction

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceBarreP`.direction`

Description

Propriété : indique la direction de remplissage de la barre de progrès. La valeur par défaut est `"right"`.

Exemple

Le code suivant indique à la barre de progrès de se remplir de droite à gauche.

```
barreP.direction = "left";
```

ProgressBar.indeterminate

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
occurrenceBarreP.indeterminate
```

Description

Propriété : valeur booléenne indiquant si la barre de progrès a un remplissage rayé et une source de chargement de taille inconnue (`true`), ou un remplissage uni et une source de chargement de taille connue (`false`).

Exemple

Le code suivant crée une barre de progrès déterminée avec un remplissage uni qui se déplace de la gauche vers la droite :

```
barreP.direction = "right";
barreP.indeterminate = false;
```

ProgressBar.label

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
occurrenceBarreP.label
```

Description

Propriété : texte indiquant la progression du chargement. Cette propriété est une chaîne au format « %1 sur %2 chargés (%3%%) » ; %1 est un espace réservé pour les octets courants chargés, %2 est un espace réservé pour le total des octets chargés et %3 est un espace réservé pour le pourcentage de contenu chargé. Les caractères « %% » sont un espace réservé pour le caractère « % ». Si une valeur pour %2 est inconnue, elle est remplacée par « ?? ». Si une valeur est undefined, l'étiquette ne s'affiche pas. La valeur par défaut est `"LOADING %3%%"`

Exemple

Le code suivant définit le texte devant apparaître à côté de la barre de progrès au format « 4 fichiers chargés » :

```
barreP.label = "%1 fichiers chargés";
```

Consultez également

ProgressBar.labelPlacement

ProgressBar.labelPlacement

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceBarreP.labelPlacement

Description

Propriété : définit le placement de l'étiquette par rapport à la barre de progrès. Les valeurs possibles sont "left", "right", "top", "bottom" et "center".

Exemple

Le code suivant définit l'étiquette à afficher au-dessus de la barre de progrès :

```
barreP.label = "%1 sur %2 chargés (%3%%)";
barreP.labelPlacement = "top";
```

Consultez également

ProgressBar.label

ProgressBar.maximum

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceBarreP.maximum

Description

Propriété : la plus grande valeur de la barre de progrès lorsque la propriété ProgressBar.mode est définie sur "manual".

Exemple

Le code suivant fixe la propriété maximum au nombre total d'images d'une application Flash en cours de chargement :

```
barreP.maximum = _totalframes;
```

Consultez également

ProgressBar.minimum, ProgressBar.mode

ProgressBar.minimum

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceBarreP.minimum

Description

Propriété : la plus petite valeur de la barre de progrès lorsque la propriété ProgressBar.mode est définie sur "manual".

Exemple

Le code suivant définit la valeur minimum de la barre de progrès :

```
barreP.minimum = 0;
```

Consultez également

ProgressBar.maximum, ProgressBar.mode

ProgressBar.mode

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceBarreP.mode

Description

Propriété : mode dans lequel la barre de progrès charge le contenu. Cette valeur peut être : `"event"`, `"polled"` ou `"manual"`. Les modes les plus communément utilisés sont `"event"` et `"polled"`. Ils utilisent le paramètre `source` pour spécifier un processus de chargement qui émet des événements `progress` et `complete`, tel un composant Loader (mode event), ou expose les méthodes `getBytesLoaded` et `getsBytesTotal`, tel un objet MovieClip (mode polled). Vous pouvez également utiliser le composant ProgressBar en mode manuel en définissant manuellement les propriétés `maximum`, `minimum` et `indeterminate` ainsi que les appels à la méthode `ProgressBar.setProgress()`.

Vous devez utiliser un objet Loader comme source en mode event. En mode polled, vous pouvez utiliser comme source tout objet exposant les méthodes `getBytesLoaded()` et `getBytesTotal()`, y compris un objet personnalisé ou l'objet `_root`.

Exemple

Le code suivant définit la barre de progrès en mode event :

```
barreP.mode = "event";
```

ProgressBar.percentComplete

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceBarreP`.percentComplete`

Description

Propriété (lecture seule) : renvoie le pourcentage de progression du processus. Cette valeur est mise au plancher. La formule suivante est utilisée pour calculer le pourcentage :

```
100*(valeur-minimum)/(maximum-minimum)
```

Exemple

Le code suivant envoie la valeur de la propriété `percentComplete` au panneau de sortie :

```
trace("pourcentage terminé = " + barreP.percentComplete);
```

ProgressBar.progress

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(progress){
   ...
}
```

Usage 2 :

```
ObjetDécoute = new Object();
objetDécoute.progress = function(objetEvnt){
   ...
}
occurrenceBarreP.addEventListener("progress", objetDécoute)
```

Objet événement

Outre les propriétés d'objet événement standard, deux propriétés supplémentaires sont définies pour l'événement `ProgressBar.progress` : `current` (la valeur chargée qui est égale au total) et `total` (la valeur totale).

Description

Evénement : diffusé à l'ensemble des écouteurs enregistrés lorsque la valeur d'une barre de progrès est modifiée. Cet événement est uniquement diffusé lorsque `ProgressBar.mode` est défini sur `"manual"` ou `"polled"`.

Le premier exemple d'utilisation fait appel à un gestionnaire `on()` et doit être directement associé à une occurrence de composant ProgressBar. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de comportement. Par exemple, le code suivant, associé à l'occurrence `maBarreP`, envoie « _level0.maBarreP » au panneau de sortie :

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`occurrenceBarreP`*) distribue un événement (ici, `progress`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

Cet exemple crée un objet d'écoute, `form`, et définit un gestionnaire d'événement `progress` sur celui-ci. L'écouteur `form` est enregistré sur l'occurrence `barreP` dans la dernière ligne de code. Lorsque l'événement `progress` est déclenché, `barreP` diffuse l'événement à l'écouteur `form` qui appelle la fonction de rappel `progress` de la manière suivante :

```
var form:Object = new Object();
form.progress = function(ObjEvnt){
```

```
    // objEvnt.target est le composant ayant généré l'événement change,
    // c.-à-d., la barre de progrès.
    trace("Valeur changée en" + ObjEvnt.target.value);
}
barreP.addEventListener("progress", form);
```

Consultez également

`UIEventDispatcher.addEventListener()`

ProgressBar.setProgress()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceBarreP`.setProgress(`*completed*`, `*total*`)`

Paramètres

completed nombre indiquant le niveau progression atteint. Vous pouvez utiliser les propriétés `ProgressBar.label` et `ProgressBar.conversion` pour afficher le nombre sous forme de pourcentage ou dans l'unité de votre choix, en fonction de la source de la barre de progrès.

total nombre indiquant la progression totale devant être effectuée pour atteindre 100 pourcent.

Renvoie

nombre indiquant le niveau de progression atteint.

Description

Méthode : définit l'état de la barre pour refléter la progression effectuée lorsque la propriété `ProgressBar.mode` est définie sur `"manual"`. Vous pouvez appeler cette méthode pour que la barre reflète l'état d'un processus autre que le chargement. L'argument completed est affecté à une propriété value et un argument total est affecté à la propriété maximum. La propriété minimum n'est pas altérée.

Exemple

Le code suivant appelle la méthode `setProgress()` en fonction de la progression du scénario d'une animation Flash :

```
barreP.setProgress(_currentFrame, _totalFrames);
```

ProgressBar.source

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
occurrenceBarreP.source
```

Description

Propriété : référence à l'occurrence devant être chargée et dont le processus de chargement sera affiché. Le contenu en chargement doit émettre un événement `progress` à partir duquel les valeurs courantes et totales sont récupérées. Cette propriété est uniquement utilisée lorsque `ProgressBar.mode` est défini sur `"event"` ou `"polled"`. La valeur par défaut est undefined.

Vous pouvez utiliser la barre de progrès avec du contenu dans une application, y compris `_root`.

Exemple

L'exemple suivant configure l'occurrence `BarreP` afin qu'elle affiche la progression du chargement d'un composant Loader dont le nom d'instance correspond à `loader` :

```
barreP.source = loader;
```

Consultez également

`ProgressBar.mode`

ProgressBar.value

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
occurrenceBarreP.value
```

Description

Propriété (lecture seule) : indique le niveau de progression atteint. Cette propriété est un nombre compris entre la valeur de `ProgressBar.minimum` et `ProgressBar.maximum`. La valeur par défaut est 0.

Composant RadioButton

Le composant RadioButton vous permet d'obliger un utilisateur à faire un choix unique parmi plusieurs possibilités. Le composant RadioButton doit être utilisé dans un groupe comprenant au moins deux occurrences RadioButton. Un seul membre du groupe peut être choisi à tout moment donné. La sélection d'un bouton radio dans un groupe désélectionne le bouton jusqu'à alors sélectionné dans le groupe. Vous pouvez définir le paramètre `groupName` pour indiquer le groupe auquel appartient un bouton radio.

Un bouton radio peut être activé ou désactivé. Lorsqu'un utilisateur utilise la tabulation dans un groupe de boutons radio, seul le bouton radio sélectionné reçoit le focus. Un utilisateur peut utiliser les touches fléchées pour modifier le focus à l'intérieur du groupe. Lorsqu'il est désactivé, un bouton radio ne reçoit pas les informations provenant du clavier ou de la souris.

Un groupe de composant RadioButton reçoit le focus si vous cliquez dessus ou utilisez la tabulation pour l'ouvrir. Lorsqu'un groupe RadioButton a le focus, vous pouvez utiliser les touches suivantes pour le contrôler :

Touche	Description
Haut/Droite	La sélection se déplace vers le bouton radio précédent dans le groupe de boutons radio.
Bas/Gauche	La sélection se déplace vers le bouton radio suivant dans le groupe de boutons radio.
Tab	Déplace le focus du groupe de boutons radio vers le composant suivant.

Pour plus d'informations sur le contrôle du focus, consultez *Création de la navigation personnalisée du focus*, page 25 ou *Classe FocusManager*, page 103.

Un aperçu en direct de chaque occurrence RadioButton sur la scène reflète les modifications effectuées sur les paramètres dans le panneau de l'inspecteur des propriétés ou des composants lors de la programmation. Cependant, l'exclusion mutuelle de la sélection ne s'affiche pas dans l'aperçu en direct. Si vous définissez le paramètre sélectionné sur true pour deux boutons radios dans un même groupe, ils apparaissent tous deux comme étant sélectionnés même si seule la dernière occurrence créée apparaîtra comme étant sélectionnée à l'exécution. Pour plus d'informations, consultez *Paramètres RadioButton*, page 180.

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

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

Vous n'activez l'accessibilité d'un composant qu'une fois, quel que soit son nombre d'occurrences. Pour plus d'informations, consultez « Création de contenu accessible », dans le guide Utilisation de Flash de l'aide. Il vous faudra parfois mettre votre système d'aide à jour pour obtenir ces informations.

Utilisation du composant RadioButton

Un bouton radio est une partie essentielle de tout formulaire ou application web. Vous pouvez utiliser les boutons radio partout où vous souhaitez qu'un utilisateur opte pour un choix dans une liste d'options. Par exemple, utilisez des boutons radio dans un formulaire pour demander quelle carte de crédit un utilisateur choisit pour effectuer le paiement.

Paramètres RadioButton

Les paramètres de programmation suivants peuvent être définis pour chaque occurrence de composant RadioButton dans le panneau de l'inspecteur des propriétés ou des composants :

label définit la valeur du texte sur le bouton. La valeur par défaut est Radio Button.

data correspond aux données associées au bouton radio. Il n'y a pas de valeur par défaut.

groupName est le nom du groupe du bouton radio. La valeur par défaut est radioGroup.

selected définit la valeur initiale du bouton radio sur sélectionné (true) ou non sélectionné (false). Un bouton radio sélectionné affiche un point. Seul un bouton radio dans un groupe peut avoir une valeur sélectionnée définie sur true. Si, dans un groupe, plusieurs boutons radio sont définis sur true, le dernier bouton radio à être instancié est sélectionné. La valeur par défaut est false.

labelPlacement oriente le texte d'étiquette sur le bouton. Ce paramètre peut avoir l'un des quatre paramètres suivants : gauche, droite, haut ou bas ; la valeur par défaut est droite. Pour plus d'informations, consultez `radioButton.labelPlacement`.

ActionScript vous permet de définir des options supplémentaires pour les occurrences RadioButton à l'aide des méthodes, propriétés et événements de la classe RadioButton. Pour plus d'informations, consultez Classe RadioButton.

Création d'une application avec le composant RadioButton

La procédure suivante explique comment ajouter des composants RadioButton à une application lors de la programmation. Dans cet exemple, les boutons radio sont utilisés pour présenter une question à laquelle on ne peut répondre que par oui ou par non, « Etes-vous joueur ? ». Les données du groupe radio sont affichées dans un composant TextArea avec `leVerdict` comme nom d'occurrence.

Pour créer une application avec le composant RadioButton, effectuez les opérations suivantes :

1. Faites glisser deux composants RadioButton du panneau Composants jusqu'à la scène.
2. Sélectionnez l'un des boutons radio et effectuez les opérations suivantes dans le panneau Inspecteur de composants :
   - Entrez Oui pour le paramètre label.
   - Entrez Joueur pour le paramètre data.
3. Sélectionnez l'autre bouton radio et, dans le panneau Inspecteur des composants, effectuez les opérations suivantes :
   - Entrez Non pour le paramètre label.
   - Entrez Anti-joueur pour le paramètre data.
4. Choisissez l'image 1 dans le scénario, ouvrez le panneau Actions et saisissez le code suivant :

```
joueurListener = new Object();
joueurListener.click = function (evt){
     leVerdict.text = evt.target.selection.data
}
radioGroup.addEventListener("click", joueurListener);
```

La dernière ligne de code ajoute un gestionnaire d'événement `click` au groupe de boutons radio `radioGroup`. Le gestionnaire définit la propriété `text` de l'occurrence de composant TextArea `leVerdict` à la valeur de la propriété `data` du bouton radio sélectionné dans le groupe de boutons radio `radioGroup`. Pour plus d'informations, consultez `RadioButton.click`.

Personnalisation du composant RadioButton

Vous pouvez transformer un composant RadioButton horizontalement et verticalement au cours de la programmation et à l'exécution. Lors de la création, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez la méthode `setSize()` (consultez *UIObject.setSize()*, page 252).

Le cadre de délimitation d'un composant RadioButton est invisible et désigne également la zone active du composant. Si vous augmentez la taille du composant, vous augmentez également la taille de la zone active.

Si la dimension du cadre de délimitation du composant est trop petite pour l'étiquette du composant, celle-ci sera rognée.

Utilisation de styles avec le composant RadioButton

Vous pouvez définir les propriétés des styles afin de modifier l'apparence d'un bouton radio. Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, consultez *Utilisation des styles pour personnaliser la couleur et le texte des composants*, page 27.

Un composant RadioButton utilise les styles de halo suivants :

Style	Description
`themeColor`	Arrière-plan d'un composant. Il s'agit du seul style de couleur qui n'hérite pas de sa valeur. "haloGreen", "haloBlue" et "haloOrange" sont des valeurs possibles.
`color`	Texte d'une étiquette de composant.
`disabledColor`	Couleur désactivée pour du texte.
`fontFamily`	Nom de police pour du texte.
`fontSize`	Taille en points pour la police.
`fontStyle`	Style de la police : "normal" ou "italic".
`fontWeight`	Epaisseur de la police : "normal" ou "bold".

Utilisation d'enveloppes avec le composant RadioButton

Vous pouvez envelopper le composant RadioButton au cours de la programmation en modifiant ses symboles dans la bibliothèque. Les enveloppes du composant RadioButton se trouvent dans le dossier suivant de la bibliothèque de HaloTheme.fla ou SampleTheme.fla : Flash UI Components 2/Themes/MMDefault/RadioButton Assets/States. Pour plus d'informations, consultez *A propos de l'application des enveloppes aux composants*, page 37.

Si un bouton radio est activé mais n'est pas sélectionné, il affiche son état survolé lorsque l'utilisateur place le pointeur de la souris au-dessus du bouton. Lorsque l'utilisateur clique sur un bouton radio non sélectionné, le bouton radio reçoit le focus d'entrée et affiche son état false enfoncé. Lorsque l'utilisateur relâche le bouton de la souris, le bouton radio affiche son état true et le bouton radio sélectionné précédemment dans le groupe revient à son état false. Si l'utilisateur éloigne le pointeur d'un bouton radio tout en appuyant sur le bouton de la souris, l'apparence du bouton revient à son état false et conserve le focus d'entrée.

Si un bouton radio ou un groupe de boutons radio est désactivé, il affiche l'état désactivé, quelle que soit l'interaction de l'utilisateur.

Si vous utilisez la méthode `UIObject.createClassObject()` pour créer de manière dynamique une occurrence du composant RadioButton, vous pouvez aussi utiliser de manière dynamique une enveloppe pour le composant. Pour utiliser de manière dynamique une enveloppe pour un composant RadioButton, transmettez les propriétés de l'enveloppe à la méthode `UIObject.createClassObject()`. Pour plus d'informations, consultez *A propos de l'application des enveloppes aux composants*, page 37. Les propriétés d'enveloppe indiquent le symbole à utiliser pour afficher un composant.

Un composant RadioButton utilise les propriétés d'enveloppe suivantes :

Nom	Description
`falseUpIcon`	L'état non coché. La valeur par défaut est `radioButtonFalseUp`.
`falseDownIcon`	L'état non coché enfoncé. La valeur par défaut est `radioButtonFalseDown`.
`falseOverIcon`	L'état survolé non coché. La valeur par défaut est `radioButtonFalseOver`.
`falseDisabledIcon`	L'état désactivé non coché. La valeur par défaut est `radioButtonFalseDisabled`.
`trueUpIcon`	L'état coché. La valeur par défaut est `radioButtonTrueUp`.
`trueDisabledIcon`	L'état désactivé coché. La valeur par défaut est `radioButtonTrueDisabled`.

Classe RadioButton

Héritage UIObject > UIComponent > SimpleButton > Bouton > RadioButton

Nom du logiciel ActionScript mx.controls.RadioButton

Les propriétés de la classe RadioButton vous permettent de créer au cours de l'exécution une étiquette de texte et de la disposer par rapport au bouton radio. Vous pouvez aussi affecter des valeurs de données aux boutons radio, les affecter à des groupes et les sélectionner en fonction de la valeur de données ou du nom de l'occurrence.

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

Le composant RadioButton utilise FocusManager pour annuler le rectangle de focus de Flash Player et dessiner un rectangle de focus personnalisé avec des angles arrondis. Pour en savoir plus sur la navigation du focus, consultez *Création de la navigation personnalisée du focus*, page 25.

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

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

Remarque : Le code suivant renvoie la valeur undefined : `trace(monOccurrenceDeBoutonRadio.version);`.

Méthodes de la classe RadioButton

Hérite de toutes les méthodes de la Classe UIObject, Classe UIComponent, de SimpleButton et de la Classe Button.

Propriétés de la classe RadioButton

Propriété	Description
`RadioButton.data`	La valeur associée à une occurrence de bouton radio.
`RadioButton.groupName`	Le nom du groupe d'un groupe de boutons radio ou d'une occurrence de bouton radio.
`RadioButton.label`	Le texte affiché à côté d'un bouton radio.
`radioButton.labelPlacement`	L'orientation du texte de l'étiquette par rapport à un bouton radio.
`RadioButton.selected`	Définit l'état de l'occurrence de bouton radio sur selected et désélectionne le bouton radio sélectionné précédemment.
`RadioButton.selectedData`	Sélectionne le bouton radio dans un groupe de boutons radio avec la valeur de données spécifiée.
`RadioButton.selection`	Une référence au bouton radio actuellement sélectionné dans un groupe de boutons radio.

Hérite de toutes les méthodes de la Classe UIObject, Classe UIComponent, de SimpleButton et de la Classe Button.

Evénements de la classe RadioButton

Evénement	Description
`RadioButton.click`	Déclenché lorsque l'on appuie sur le bouton de la souris au-dessus d'une occurrence de bouton.

Hérite de tous les événements de la Classe UIObject, Classe UIComponent, de SimpleButton et de la Classe Button.

RadioButton.click

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.click = function(eventObject){
  ...
}
GroupeDeBoutonsRadio.addEventListener("click", objetDécoute)
```

Description

Evénement : diffuse à tous les écouteurs enregistrés lorsque l'utilisateur clique à l'aide de la souris (bouton de la souris enfoncé et relâché) sur le bouton radio ou si le bouton radio est sélectionné en utilisant les boutons fléchés. L'événement est aussi diffusé si l'on appuie sur la barre d'espace ou sur les boutons fléchés lorsqu'un groupe de boutons radio a le focus mais aucun des boutons radio du groupe n'est sélectionné.

Le premier exemple d'utilisation recourt à un gestionnaire `on()` et doit être directement associé à une occurrence du composant RadioButton. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de comportement. Par exemple, le code suivant, associé au bouton radio `monBoutonRadio`, envoie "_level0.monBoutonRadio" au panneau de sortie :

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`OccurrenceDeBoutonRadio`*) distribue un événement (ici, `click`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. L'objet événement a un jeu de propriétés contenant des informations sur l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

Cet exemple, écrit sur une image du scénario, envoie un message au panneau de sortie lorsqu'on clique sur un bouton radio dans `radioGroup`. La première ligne de code crée un objet d'écoute intitulé `form`. La deuxième ligne définit une fonction pour l'événement `click` sur l'objet d'écoute. La fonction comporte une action `trace` qui utilise l'objet événement automatiquement transmis à cette fonction, ici `objEvnt`, pour générer un message. La propriété `target` d'un objet événement est le composant qui a généré l'événement. Vous pouvez accéder aux propriétés de l'occurrence à partir de la propriété `target` (dans cet exemple on accède à la propriété `RadioButton.selection`). La dernière ligne appelle la méthode `UIEventDispatcher.addEventListener()` à partir de `radioGroup` et transmet l'événement `click` et l'objet d'écoute `form` comme paramètres de la manière suivante :

```
form = new Object();
form.click = function(objEvnt){
  trace("L'occurrence radio sélectionnée est" + objEvnt.target.selection);
```

```
}
radioGroup.addEventListener("click", form);
```

Le code suivant envoie aussi un message au panneau de sortie lorsqu'on clique sur `OccurrenceDeBoutonRadio`. Le gestionnaire `on()` doit être directement associé à `OccurrenceDeBoutonRadio` de la manière suivante :

```
on(click){
  trace("composant bouton radio cliqué");
}
```

RadioButton.data

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`OccurrenceDeBoutonRadio``.data`

Description

Propriété : spécifie les données à associer à une occurrence de bouton radio. La définition de cette propriété annule la valeur du paramètre de données définie au cours de la programmation dans le panneau de l'inspecteur des propriétés ou des composants. La propriété `data` peut être n'importe quel type de données.

Exemple

L'exemple suivant affecte la valeur de données `"#FF00FF"` à l'occurrence de bouton radio `radioOne` :

```
radioOne.data = "#FF00FF";
```

RadioButton.groupName

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`OccurrenceDeBoutonRadio``.groupName`
`GroupeDeBoutonsRadio``.groupName`

Description

Propriété : définit le nom du groupe d'une occurrence de bouton radio ou d'un groupe de boutons radio. Vous pouvez utiliser cette propriété afin d'obtenir ou de définir un nom de groupe pour une occurrence de bouton radio ou pour un groupe de boutons radio. L'appel de cette méthode annule la valeur du paramètre groupName définie au cours de la programmation. La valeur par défaut est `"radioGroup"`.

Exemple

L'exemple suivant définit le nom du groupe d'une occurrence de bouton radio sur `"choixCouleur"`, puis transforme le nom en `"choixTaille"`. Pour tester cet exemple, placez un bouton radio sur la scène avec le nom d'occurrence `monBoutonRadio` et saisissez le code suivant sur la première image :

```
monBoutonRadio.groupName = "choixCouleur";
trace(monBoutonRadio.groupName);
choixCouleur.groupName = "choixTaille";
trace(choixCouleur.groupName);
```

RadioButton.label

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`OccurrenceDeBoutonRadio``.label`

Description

Propriété : spécifie le texte de l'étiquette du bouton radio. Par défaut, l'étiquette apparaît à droite du bouton radio. L'appel de cette méthode annule le paramètre label spécifié au cours de la programmation. Si le texte de l'étiquette est trop long et ne rentre pas dans le cadre de délimitation du composant, le texte est rogné.

Exemple

L'exemple suivant définit la propriété de l'étiquette de l'occurrence `radioButton` :

```
radioButton.label = "Supprimer de la liste";
```

radioButton.labelPlacement

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`OccurrenceDeBoutonRadio``.labelPlacement`
`groupeDeBoutonsRadio``.labelPlacement`

Description

Propriété : chaîne indiquant la position de l'étiquette par rapport à un bouton radio. Pour pouvez définir cette propriété pour une occurrence individuelle ou pour un groupe de boutons radio. Si vous définissez la propriété pour un groupe, l'étiquette est placée à l'endroit approprié pour chaque bouton radio du groupe.

Quatre valeurs sont possibles :

- `"right"` Le bouton radio est placé dans le coin supérieur gauche du cadre de délimitation. L'étiquette est placée à droite du bouton radio.
- `"left"` Le bouton radio est placé dans le coin supérieur droit du cadre de délimitation. L'étiquette est placée à gauche du bouton radio.
- `"bottom"` L'étiquette est placée sous le bouton radio. Le bouton radio et l'étiquette sont centrés horizontalement et verticalement. Si la dimension du cadre de délimitation du bouton radio est trop réduite, l'étiquette sera rognée.
- `"top"` L'étiquette est placée au-dessus du bouton radio. Le bouton radio et l'étiquette sont centrés horizontalement et verticalement. Si la dimension du cadre de délimitation du bouton radio est trop réduite, l'étiquette sera rognée.

Exemple

Le code suivant place l'étiquette à gauche de chaque bouton radio dans `radioGroup`:

```
radioGroup.labelPlacement = "left";
```

RadioButton.selected

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
OccurrenceDeBoutonRadio.selected
GroupeDeBoutonsRadio.selected
```

Description

Propriété : valeur booléenne qui définit l'état du bouton radio sur sélectionné (`true`) et désélectionne le bouton radio sélectionné précédemment ou définit l'état du bouton radio sur désélectionné (`false`).

Exemple

La première ligne de code définit l'occurrence `mcButton` sur `true`. La deuxième ligne de code renvoie la valeur de la propriété sélectionnée comme suit :

```
mcButton.selected = true;
trace(mcButton.selected);
```

RadioButton.selectedData

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

monGroupeDeBoutonsRadio.selectedData

Description

Propriété : sélectionne le bouton radio avec la valeur de données spécifiée et désélectionne le bouton radio sélectionné précédemment. Si la propriété `data` n'est pas spécifiée pour une occurrence sélectionnée, la valeur de l'étiquette de l'occurrence sélectionnée est sélectionnée et renvoyée. La propriété `selectedData` peut être n'importe quel type de données.

Exemple

L'exemple suivant sélectionne le bouton radio avec la valeur `"#FF00FF"` à partir du groupe `colorGroup` et envoie la valeur au panneau de sortie :

```
colorGroup.selectedData = "#FF00FF";
trace(colorGroup.selectedData);
```

RadioButton.selection

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeBoutonRadio.selection
GroupeDeBoutonsRadio.selection

Description

Propriété : se comporte de manière différente si vous obtenez ou définissez la propriété. Si vous obtenez la propriété, elle renvoie la référence objet du bouton radio actuellement sélectionné dans un groupe de boutons radio. Si vous définissez la propriété, elle sélectionne le bouton radio spécifié (transmis comme référence objet) dans un groupe de boutons radio et désélectionne le bouton radio précédemment sélectionné.

Exemple

L'exemple suivant sélectionne le bouton radio avec le nom d'occurrence `color1` et envoie le nom de son occurrence au panneau de sortie :

```
colorGroup.selection = color1;
trace(colorGroup.selection._name)
```

Composant RDBMSResolver

Pour des informations actualisées sur cette fonction, cliquez sur le bouton Mettre à jour dans la partie supérieure de l'onglet Aide.

Interface RemoteProcedureCall

Pour des informations actualisées sur cette fonction, cliquez sur le bouton Mettre à jour dans la partie supérieure de l'onglet Aide.

Classe Screen

Pour des informations actualisées sur cette fonction, cliquez sur le bouton Mettre à jour dans la partie supérieure de l'onglet Aide.

Composant ScrollPane

Le composant Scroll Pane affiche des clips, des fichiers JPEG et SWF dans une zone défilante. Vous pouvez activer les barres de défilement de façon à afficher les images dans une zone limitée. Vous pouvez afficher du contenu chargé à partir d'un emplacement local ou d'Internet. Vous pouvez définir le contenu du panneau défilant à la fois au cours de la programmation et au cours de l'exécution en utilisant ActionScript.

Une fois que le panneau défilant a le focus, si le contenu du panneau défilant présente des arrêts de tabulation valides, ces marqueurs reçoivent le focus. Après le dernier arrêt de tabulation dans le contenu, le focus passe au composant suivant. Les barres de défilement horizontale et verticale dans le panneau défilant ne reçoivent jamais le focus.

Une occurrence de ScrollPane reçoit le focus si un utilisateur clique dessus ou utilise les tabulations. Lorsqu'une occurrence de ScrollPane a le focus, vous pouvez utiliser les touches suivantes pour contrôler le contenu :

Touche	Description
Bas	Le contenu se déplace d'une ligne de défilement verticale vers le haut.
Fin	Le contenu se déplace en bas du panneau défilant.
Gauche	Le contenu se déplace d'une ligne de défilement horizontale vers la droite.
Origine	Le contenu se déplace en haut du panneau défilant.
Pg. Suiv.	Le contenu se déplace d'une page de défilement verticale vers le haut.
Pg. Préc.	Le contenu se déplace d'une page de défilement verticale vers le bas.
Droite	Le contenu se déplace d'une ligne de défilement horizontale vers la gauche.
Haut	Le contenu se déplace d'une ligne de défilement verticale vers le bas.

Pour plus d'informations sur le contrôle du focus, consultez *Création de la navigation personnalisée du focus*, page 25 ou *Classe FocusManager*, page 103.

Un aperçu en direct de chaque occurrence de ScrollPane reflète les changements apportés dans le panneau de l'inspecteur des propriétés ou des composants au cours de la programmation.

Utilisation du composant ScrollPane

Vous pouvez utiliser un panneau défilant pour afficher le contenu qui ne rentre pas dans la zone dans laquelle il est chargé. Par exemple, si vous devez afficher une image de grande taille mais que vous avez peu de place dans une application, vous pouvez la charger dans un panneau défilant.

Vous pouvez définir le paramètre `scrollDrag` d'un panneau défilant sur `true` afin de permettre aux utilisateurs de déplacer le contenu à l'intérieur du panneau ; un curseur en forme de main apparaît sur le contenu. Contrairement à la plupart des autres composants, les événements sont diffusés lorsque le bouton de la souris est enfoncé et continuent à diffuser jusqu'à ce que l'utilisateur relâche le bouton de la souris. Si le contenu d'un panneau défilant comporte des arrêts de tabulation valides, vous devez définir `scrollDrag` sur `false` sinon chaque interaction de la souris avec les contenus invoquera le déplacement par défilement.

Paramètres du composant ScrollPane

Voici les paramètres de programmation que vous pouvez définir pour chaque occurrence du composant ScrollPane dans le panneau de l'inspecteur des propriétés ou des composants :

contentPath indique le contenu à charger dans le panneau défilant. Cette valeur peut être un chemin relatif vers un fichier local SWF ou JPEG, ou un chemin relatif ou absolu vers un fichier sur Internet. Il peut aussi s'agir de l'identificateur de liaison du symbole d'un clip dans la bibliothèque, défini sur Exporter pour ActionScript.

hLineScrollSize indique de combien d'unités se déplace une barre de défilement horizontale à chaque fois qu'un bouton fléché est enfoncé. La valeur par défaut est 5.

hPageScrollSize indique de combien d'unités se déplace une barre de défilement horizontale à chaque fois que le rail est enfoncé. La valeur par défaut est 20.

hScrollPolicy affiche les barres de défilement horizontales. La valeur peut être "on", "off" ou "auto". La valeur par défaut est "auto"

scrollDrag est une valeur booléenne qui permet (true) ou ne permet pas (false) à l'utilisateur de faire défiler le contenu à l'intérieur du panneau défilant. La valeur par défaut est false.

vLineScrollSize indique de combien d'unités se déplace une barre de défilement verticale à chaque fois qu'un bouton fléché est enfoncé. La valeur par défaut est 5.

hPageScrollSize indique de combien d'unités se déplace une barre de défilement verticale à chaque fois que le rail est enfoncé. La valeur par défaut est 20.

vScrollPolicy affiche les barres de défilement verticales. La valeur peut être "on", "off" ou "auto". La valeur par défaut est "auto"

Vous pouvez rédiger du code ActionScript pour contrôler ces options et d'autres options des composants ScrollPane en utilisant les propriétés, méthodes et événements ActionScript. Pour plus d'informations, consultez Classe ScrollPane.

Création d'une application avec le composant ScrollPane

La procédure suivante explique comment ajouter un composant ScrollPane à une application au cours de la programmation. Dans cet exemple, le panneau défilant charge un fichier SWF contenant un logo.

Pour créer une application avec le composant ScrollPane, procédez ainsi :

1. Faites glisser un composant du panneau Composants jusqu'à la scène.
2. Dans l'inspecteur des propriétés, saisissez le nom d'occurrence **monPanneauDéfilant**.
3. Dans l'inspecteur des propriétés, saisissez **logo.swf** comme paramètre contentPath.
4. Choisissez l'image 1 dans le scénario, ouvrez le panneau Actions et saisissez le code suivant :

```
scrollListener = new Object();
scrollListener.scroll = function (evt){
  txtPosition.text = monPanneauDéfilant.vPosition;
}
monPanneauDéfilant.addEventListener("scroll", scrollListener);

completeListener = new Object;
completeListener.complete = function() {
  trace("logo.swf a fini le chargement.");
}
monPanneauDéfilant.addEventListener("complete", completeListener);
```

Le premier bloc de code est un gestionnaire d'événement `scroll` sur l'occurrence `monPanneauDéfilant` qui affiche la valeur de la propriété `vPosition` dans une occurrence ChampDeTexte appelée `txtPosition`, qui a déjà été placée sur la scène. Le deuxième bloc de code crée un gestionnaire d'événement pour l'événement `complete` qui envoie un message au panneau de sortie.

Personnalisation du composant ScrollPane

Vous pouvez transformer un composant ScrollPane horizontalement et verticalement à la fois au cours de la programmation et de l'exécution. Lors de la création, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez la méthode `setSize()` (consultez `UIObject.setSize()`) ou toute propriété et méthode applicables de la classe ScrollPane. Pour plus d'informations, consultez Classe ScrollPane. Si le panneau défilant n'est pas assez grand, il se peut que le contenu ne s'affiche pas correctement.

Le panneau défilant définit le coin supérieur gauche comme le point d'alignement de son contenu.

Lorsque la barre de défilement horizontale est désactivée, la barre de défilement verticale s'affiche de haut en bas sur le côté droit du panneau défilant. Lorsque la barre de défilement verticale est désactivée, la barre de défilement horizontale s'affiche de gauche à droite en bas du panneau défilant. Vous pouvez aussi désactiver ces deux barres.

Lorsque le panneau défilant est redimensionné, les boutons conservent la même taille tandis que le rail et le curseur de défilement sont agrandis ou réduits et leurs zones réactives sont redimensionnées.

Utilisation des styles avec le composant ScrollPane

Le composant ScrollPane ne gère pas les styles, contrairement aux barres de défilement qu'il utilise. Pour plus d'informations, consultez *Utilisation des styles avec le composant ScrollPane*, page 192.

Utilisation des enveloppes avec le composant ScrollPane

Le composant ScrollPane ne dispose d'aucune enveloppe contrairement aux barres de défilement qu'il utilise. Pour plus d'informations, consultez *Utilisation des enveloppes avec le composant ScrollPane*, page 193.

Classe ScrollPane

Héritage UIObject > UIComponent > View > ScrollView > ScrollPane

Espace de nom de classe ActionScript mx.containers.ScrollPane

Les propriétés de la classe ScrollPane vous permettent de définir le contenu, de contrôler la progression du chargement et de régler la quantité à faire défiler au cours de l'exécution.

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

Vous pouvez définir la propriété `scrollDrag` d'un panneau défilant sur `true` afin de permettre aux utilisateurs de déplacer le contenu à l'intérieur du panneau ; un curseur en forme de main apparaît sur le contenu. Contrairement à la plupart des autres composants, les événements sont diffusés lorsque le bouton de la souris est enfoncé et continuent à diffuser jusqu'à ce que l'utilisateur relâche le bouton de la souris. Si le contenu d'un panneau défilant comporte des arrêts de tabulation valides, vous devez définir `scrollDrag` sur `false` sinon chaque interaction de la souris avec les contenus invoquera le déplacement par défilement.

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

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

Remarque : Le code suivant renvoie la valeur undefined : `trace(monOccurrenceDePanneauDéfilant.version);`.

Méthodes de la classe ScrollPane

Méthode	Description
`ScrollPane.getBytesLoaded()`	Renvoie le nombre d'octets du contenu chargé.
`ScrollPane.getBytesTotal()`	Renvoie le nombre total d'octets du contenu à charger.
`ScrollPane.refreshPane()`	Recharge le contenu du panneau défilant.

Hérite de toutes les méthodes de la Classe UIObject et de la Classe UIComponent.

Propriétés de la classe ScrollPane

Méthode	Description
ScrollPane.content	Référence au contenu chargé dans le panneau défilant.
ScrollPane.contentPath	Une URL absolue ou relative du fichier SWF ou JPEG à charger dans le panneau défilant
ScrollPane.hLineScrollSize	La quantité de contenu à faire défiler horizontalement lorsqu'un bouton fléché est enfoncé.
ScrollPane.hPageScrollSize	La quantité de contenu à faire défiler horizontalement lorsque le rail est enfoncé.
ScrollPane.hPosition	La position horizontale (en pixels) dans le panneau défilant.
ScrollPane.hScrollPolicy	L'état de la barre de défilement horizontale. Elle peut être toujours activée (`"on"`), toujours désactivée (`"off"`) ou disponible en fonction des besoins (`"auto"`). La valeur par défaut est `"auto"`.
ScrollPane.scrollDrag	Indique si le défilement a lieu (`true`) ou non (`false`) lorsqu'un utilisateur appuie sur un bouton et fait glisser un élément à l'intérieur du panneau défilant. La valeur par défaut est `false`.
ScrollPane.vLineScrollSize	La quantité de contenu à faire défiler verticalement lorsqu'un bouton fléché est enfoncé.
ScrollPane.vPageScrollSize	La quantité de contenu à faire défiler verticalement lorsque le rail est enfoncé.
ScrollPane.vPosition	La position verticale (en pixels) dans le panneau défilant.
ScrollPane.vScrollPolicy	L'état de la barre de défilement verticale. Elle peut être toujours activée (`"on"`), toujours désactivée (`"off"`) ou disponible en fonction des besoins (`"auto"`). La valeur par défaut est `"auto"`.

Hérite de toutes les propriétés de la Classe UIObject et de la Classe UIComponent.

Evénements de la classe ScrollPane

Méthode	Description
ScrollPane.complete	Diffusion lorsque le contenu du panneau défilant est chargé.
ScrollPane.progress	Diffusion lorsque le contenu du panneau défilant est en cours de chargement.
ScrollPane.scroll	Diffusion lorsque la barre de défilement est enfoncée.

Hérite de tous les événements de la Classe UIObject et de la Classe UIComponent.

ScrollPane.complete

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(complete){
   ...
}
```

Usage 2 :

```
ObjetDécoute = new Object();
ObjetDécoute.complete = function(objetEvnt){
   ...
}
monPanneauDéfilant.addEventListener("complete", objetDécoute)
```

Description

Evénement : diffuse à tous les écouteurs enregistrés une fois le contenu chargé.

Le premier exemple d'utilisation recourt à un gestionnaire `on()` et doit être directement associé à une occurrence du composant ScrollPane. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de comportement. Par exemple, le code suivant, associé à l'occurrence du composant `monComposantPanneauDéfilant`, envoie « _level0.monComposantPanneauDéfilant » au panneau de sortie :

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`OccurrenceDePanneauDéfilant`*) distribue un événement (ici, `complete`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

L'exemple suivant crée un objet d'écoute avec un gestionnaire d'événement `complete` pour l'occurrence de `ScrollPane` :

```
form.complete = function(ObjEvnt){
  // insérez le code afin de gérer l'événement
}
scrollPane.addEventListener("complete",form);
```

ScrollPane.content

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

monPanneauDéfilant.content

Description

Propriété (lecture seule) : référence au contenu du panneau défilant. La valeur est undefined jusqu'à ce que le chargement commence.

Exemple

Cet exemple définit la variable `mcLoaded` sur la valeur de la propriété `content` :

```
var mcLoaded = scrollPane.content;
```

Consultez également

ScrollPane.contentPath

ScrollPane.contentPath

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDePanneauDéfilant.contentPath

Description

Propriété : chaîne qui indique une URL absolue ou relative du fichier SWF ou JPEG à charger dans le panneau défilant. Un chemin relatif doit être relatif au fichier SWF chargeant le contenu.

Si vous chargez le contenu en utilisant une URL relative, le contenu chargé doit être relatif à l'emplacement du fichier SWF contenant le panneau défilant. Par exemple, une application utilisant un composant ScrollPane qui réside dans le répertoire /scrollpane/nav/example.swf pourrait charger les contenus à partir du répertoire /scrollpane/content/flash/logo.swf avec la propriété contentPath suivante : `"../content/flash/logo.swf"`

Exemple

L'exemple suivant dit au panneau défilant d'afficher les contenus d'une image à partir d'Internet :

```
scrollPane.contentPath ="http://imagecache2.allposters.com/images/43/
  033_302.jpg";
```

Dans l'exemple suivant, le panneau défilant affiche le contenu d'un symbole figurant dans la bibliothèque :

```
scrollPane.contentPath ="movieClip_Name";
```

L'exemple suivant indique au panneau défilant d'afficher les contenus du fichier local « logo.swf » :

```
scrollPane.contentPath = "logo.swf";
```

Consultez également

ScrollPane.content

ScrollPane.getBytesLoaded()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDePanneauDéfilant.getBytesLoaded()

Paramètres

Aucun.

Renvoie

Le nombre d'octets chargés dans le panneau défilant.

Description

Méthode : renvoie le nombre d'octets chargés dans l'occurrence de ScrollPane. Vous pouvez appeler cette méthode à intervalles réguliers pendant le chargement du contenu afin d'en vérifier la progression.

Exemple

Cet exemple crée une occurrence de la classe ScrollPane appelée `scrollPane`. Elle définit un objet d'écoute appelé `loadListener` avec un gestionnaire d'événement `progress` qui appelle la méthode `getBytesLoaded()` afin de contribuer à déterminer la progression du chargement :

```
createClassObject(mx.containers.ScrollPane, "monPanneauDéfilant", 0);
loadListener = new Object();
loadListener.progress = function(objEvnt){
  // objEvnt.target est le composant qui a généré l'événement change
  var bytesLoaded = scrollPane.getBytesLoaded();
  var bytesTotal = scrollPane.getBytesTotal();
  var percentComplete = Math.floor(bytesLoaded/bytesTotal);

  if (percentComplete < 5 )  // le chargement vient de commencer
  {
    trace("Début du chargement du contenu à partir d'Internet");
  }
  else if(percentComplete = 50) //50 % accomplis
  {
    trace(" 50% du contenu téléchargé");
```

```
    }
}
scrollPane.addEventListener("progress", loadListener);
scrollPane.contentPath = "http://www.geocities.com/hcls_matrix/Images/
  homeview5.jpg";
```

ScrollPane.getBytesTotal()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDePanneauDéfilant.getBytesTotal()

Paramètres

Aucun.

Renvoie

Nombre.

Description

Méthode : renvoie le nombre total d'octets à charger dans l'occurrence monPanneauDéfilant .

Consultez également

ScrollPane.getBytesLoaded()

ScrollPane.hLineScrollSize

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDePanneauDéfilant.hLineScrollSize

Description

Propriété : nombre indiquant de combien de pixels se déplace le contenu lorsque le bouton fléché gauche ou droit de la barre de défilement horizontale est enfoncé. La valeur par défaut est 5.

Exemple

Cet exemple augmente l'unité de défilement horizontal en la faisant passer à 10 :

```
scrollPane.hLineScrollSize = 10;
```

ScrollPane.hPageScrollSize

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDePanneauDéfilant.hPageScrollSize

Description

Propriété : nombre indiquant de combien de pixels se déplace le contenu lorsque le rail de la barre de défilement horizontale est enfoncé. La valeur par défaut est 20.

Exemple

Cet exemple augmente l'unité de défilement horizontal de la page en la faisant passer à 30 :

```
scrollPane.hPageScrollSize = 30;
```

ScrollPane.hPosition

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDePanneauDéfilant.hPosition

Description

Propriété : la position (en pixels) de la barre de défilement horizontale. La position 0 est à gauche de la barre.

Exemple

Cet exemple définit la barre de défilement sur 20 :

```
scrollPane.hPosition = 20;
```

ScrollPane.hScrollPolicy

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDePanneauDéfilant.hScrollPolicy

Description

Propriété : détermine si la barre de défilement horizontale est toujours présente ("on"), jamais présente ("off") ou s'affiche automatiquement en fonction de la taille de l'image ("auto"). La valeur par défaut est "auto".

Exemple

Le code suivant active les barres de défilement de façon permanente :

```
scrollPane.hScrollPolicy = "on";
```

ScrollPane.progress

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(progress){
  ...
}
```

Usage 2 :

```
ObjetDécoute = new Object();
objetDécoute.progress = function(objetEvnt){
  ...
}
OccurrenceDePanneauDéfilant.addEventListener("progress", objetDécoute)
```

Description

Evénement : diffuse à l'ensemble des écouteurs enregistrés pendant le chargement du contenu. L'événement progress n'est pas toujours diffusé. L'événement `complete` peut être diffusé sans qu'aucun événement `progress` ne soit distribué. Ceci peut particulièrement se produire si le contenu chargé est un fichier local. Cet événement est déclenché lorsque le chargement commence en définissant la valeur de la propriété `contentPath`.

Le premier exemple d'utilisation recourt à un gestionnaire `on()` et doit être directement associé à une occurrence du composant ScrollPane. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de comportement. Par exemple, le code suivant, associé à l'occurrence du composant ScrollPane `monComposantSP`, envoie "_level0.monComposantSP" au panneau de sortie :

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*OccurrenceDePanneauDéfilant*) distribue un événement (ici, `progress`) qui est géré par une fonction associée à un objet d'écoute (*objetDécoute*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*objetEvnt*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

Le code suivant crée une occurrence de ScrollPane appelée `scrollPane`, puis un objet d'écoute avec un gestionnaire d'événement pour l'événement `progress` qui envoie un message au panneau de sortie à propos du nombre d'octets du contenu chargés :

```
createClassObject(mx.containers.ScrollPane, "monPanneauDéfilant", 0);
loadListener = new Object();
loadListener.progress = function(objEvnt){
  // objEvnt.target est le composant qui a généré l'événement de progression
  // dans ce cas, scrollPane
  trace("logo.swf a été chargé " + scrollPane.getBytesLoaded() + " Octets.");
  // progression du chargement de la piste
}
scrollPane.addEventListener("terminé", loadListener);
scrollPane.contentPath = "logo.swf";
```

ScrollPane.refreshPane()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDePanneauDéfilant`.refreshPane()`

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : assure le rafraîchissement du panneau défilant après le chargement du contenu. Cette méthode recharge le contenu. Vous pouvez utiliser cette méthode si, par exemple, vous avez chargé un formulaire dans un ScrollPane et qu'une propriété d'entrée (par exemple, dans un champ de texte) a été modifiée en utilisant ActionScript. Appelez `refreshPane()` pour recharger le même formulaire avec les nouvelles valeurs de propriétés d'entrée.

Exemple

L'exemple suivant permet de rafraîchir l'occurrence `sp` du panneau défilant :

```
sp.refreshPane();
```

ScrollPane.scroll

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(scroll){
  ...
}
```

Usage 2 :

```
ObjetDécoute = new Object();
objetDécoute.scroll = function(ObjetEvnt){
  ...
}
OccurrenceDePanneauDéfilant.addEventListener("scroll", objetDécoute)
```

Objet événement

Outre les propriétés standard de l'objet événement, une propriété `type` est définie pour l'événement `scroll` ; sa valeur est `"scroll"`. Il existe également une propriété `direction` pouvant avoir les valeurs `"vertical"` et `"horizontal"`.

Description

Evénement : diffuse à tous les écouteurs enregistrés lorsqu'un utilisateur appuie sur les boutons, le curseur de défilement ou le rail de la barre de défilement. Contrairement aux autres événements, l'événement `scroll` est diffusé lorsqu'un utilisateur appuie sur la barre de défilement et continue à diffuser jusqu'à ce que l'utilisateur cesse d'appuyer sur la barre de défilement.

Le premier exemple d'utilisation recourt à un gestionnaire `on()` et doit être directement associé à une occurrence du composant ScrollPane. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de comportement. Par exemple, le code suivant, associé à l'occurrence `sp`, envoie "_level0.sp" au panneau de sortie :

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*OccurrenceDePanneauDéfilant*) distribue un événement (ici, `scroll`) qui est géré par une fonction associée à un objet d'écoute (*objetDécoute*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*objetEvnt*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

L'exemple suivant crée un objet d'écoute `form` avec une fonction de rappel `scroll` enregistrée dans l'occurrence `occurrencePd` . Vous devez remplir `occurrencePd` de contenu, comme illustré ci-après :

```
occurrencePd.contentPath = "mouse3.jpg";
form = new Object();
form.scroll = function(objEvnt){
  trace("Le panneau défilant a défilé");
}
occurrencePd.addEventListener("scroll", form);
```

Consultez également

`UIEventDispatcher.addEventListener()`

ScrollPane.scrollDrag

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDePanneauDéfilant`.scrollDrag`

Description

Propriété : valeur booléenne qui indique si le défilement doit se produire lorsque l'utilisateur appuie sur le bouton de la souris et la fait glisser (`true`) ou non (`false`) dans ScrollPane. La valeur par défaut est `false`.

Exemple

Cet exemple permet d'utiliser la fonction de défilement de la souris dans le panneau de défilement :

```
scrollPane.scrollDrag = true;
```

ScrollPane.vLineScrollSize

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDePanneauDéfilant.vLineScrollSize

Description

Propriété : nombre de pixels nécessaires au déplacement de la zone d'affichage lorsque l'utilisateur clique sur la flèche vers le haut ou la flèche vers le bas d'une barre de défilement verticale. La valeur par défaut est 5.

Exemple

Ce code permet de définir le déplacement de la zone d'affichage à 10 lorsque l'utilisateur clique sur les boutons fléchés de la barre de défilement verticale :

```
scrollPane.vLineScrollSize = 10;
```

ScrollPane.vPageScrollSize

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

OccurrenceDePanneauDéfilant.vPageScrollSize

Description

Propriété : nombre de pixels nécessaires au déplacement de la zone d'affichage lorsque l'utilisateur clique sur le rail d'une barre de défilement verticale. La valeur par défaut est 20.

Exemple

Ce code permet de définir le déplacement de la zone d'affichage à 30 lorsque l'utilisateur clique sur les boutons fléchés de la barre de défilement verticale :

```
scrollPane.vPageScrollSize = 30;
```

ScrollPane.vPosition

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
OccurrenceDePanneauDéfilant.vPosition
```

Description

Propriété : position du pixel de la barre de défilement verticale. La valeur par défaut est 0.

ScrollPane.vScrollPolicy

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
OccurrenceDePanneauDéfilant.vScrollPolicy
```

Description

Propriété : détermine si la barre de défilement verticale est toujours affichée (`"on"`), jamais affichée (`"off"`) ou si elle doit s'afficher automatiquement en fonction de la taille de l'image (`"auto"`). La valeur par défaut est `"auto"`.

Exemple

Le code suivant permet un affichage systématique des barres de défilement verticales :

```
scrollPane.vScrollPolicy = "on";
```

Classe StyleManager

Espace de nom de classe ActionScript mx.styles.StyleManager

La classe StyleManager conserve une trace des styles et des couleurs d'héritage connus. Vous avez besoin de cette classe uniquement si vous créez des composants et que vous souhaitez ajouter un nouveau style ou une nouvelle couleur d'héritage.

Pour déterminer les styles d'héritage, référez-vous au site Web W3C.

Méthodes de la classe StyleManager

Méthode	Description
`StyleManager.registerColorName()`	Enregistre un nouveau nom de couleur avec StyleManager.
`StyleManager.registerColorStyle()`	Enregistre un nouveau style de couleur avec StyleManager.
`StyleManager.registerInheritingSyle()`	Enregistre un nouveau style d'héritage avec StyleManager.

StyleManager.registerColorName()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
StyleManager.registerColorName(nomCouleur, valeur)
```

Paramètres

nomCouleur Chaîne indiquant le nom de la couleur (par exemple, `"gris"`, `"grisfoncé"`, etc.).

valeur Nombre hexadécimal indiquant la couleur (par exemple, `0x808080`, `0x404040`, etc.).

Renvoie

Rien.

Description

Méthode : associe un nom de couleur à une valeur hexadécimale et l'enregistre avec StyleManager.

Exemple

Le code suivant enregistre « gris » comme le nom de la couleur qui a pour valeur hexadécimale 0x808080 :

```
StyleManager.registerColorName("gris", 0x808080 );
```

StyleManager.registerColorStyle()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
StyleManager.registerColorStyle(styleCouleur)
```

Paramètres

styleCouleur Chaîne indiquant le nom du style de la couleur (par exemple, `"highlightColor"`, `"shadowColor"`, `"disabledColor"`, etc.).

Renvoie

Rien.

Description

Méthode : ajoute un nouveau style de couleur à StyleManager.

Exemple

L'exemple suivant enregistre « highlightColor » comme un style de couleur :

```
StyleManager.registerColorStyle("highlightColor");
```

StyleManager.registerInheritingSyle()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
StyleManager.registerInheritingStyle(nomPropriété)
```

Paramètres

propertyName Chaîne indiquant le nom de la propriété du style (par exemple, `"nouvelleProp1"`, `"nouvelleProp2"`, etc.).

Renvoie

Rien.

Description

Méthode : signale cette propriété de style comme héritage. Utilisez cette méthode pour enregistrer des propriétés de styles qui ne sont pas répertoriées dans la spécification CSS. N'utilisez pas cette méthode pour transformer des propriétés de styles non-héritage en propriétés de styles d'héritage.

Exemple

L'exemple suivant enregistre `nouvelleProp1` comme un style d'héritage :

```
StyleManager.registerInheritingStyle("nouvelleProp1");
```

Classe Slide

Pour des informations actualisées sur cette fonction, cliquez sur le bouton Mettre à jour dans la partie supérieure de l'onglet Aide.

Composant TextArea

Le composant TextArea renvoie l'objet TextField ActionScript natif à la ligne. Vous pouvez utiliser les styles pour personnaliser le composant TextArea. Lorsqu'une occurrence est désactivée, son contenu s'affiche dans une couleur représentée par le style « disabledColor ». Un composant TextArea peut également être formaté en HTML ou en tant que champ de mot de passe qui masque le texte.

Un composant TextArea peut être activé ou désactivé dans une application. Lorsqu'il est désactivé, il ne reçoit pas les informations en provenance de la souris ou du clavier. Lorsqu'il est activé, il suit les mêmes règles de focus, de sélection et de navigation qu'un objet TextField ActionScript. Lorsqu'une occurrence TextArea a le focus, vous pouvez utiliser les touches suivantes pour le contrôler :

Touche	Description
Touches de flèches	Déplace le point d'insertion d'une ligne vers le haut, le bas, la gauche ou la droite.
Pg. Suiv.	Effectue un déplacement d'un écran vers le bas.
Pg. Préc.	Effectue un déplacement d'un écran vers le haut.
Maj +Tab	Place le focus sur l'objet précédent.
Tab	Place le focus sur l'objet suivant.

Pour plus d'informations sur le contrôle du focus, consultez *Création de la navigation personnalisée du focus*, page 25 ou *Classe FocusManager*, page 103.

En cours de programmation, un aperçu en direct de chaque occurrence TextArea permet de faire apparaître les modifications apportées dans les paramètres de l'inspecteur des propriétés ou dans le panneau Inspecteur de composants. Si une barre de défilement s'avère nécessaire, elle apparaît lors d'un aperçu direct, mais ne fonctionnera pas. Lors d'un aperçu direct, il n'est pas possible de sélectionner du texte et vous ne pouvez pas entrer de texte dans l'occurrence du composant sur la scène.

Lorsque vous ajoutez le composant TextArea à une application, vous pouvez utiliser le panneau Accessibilité pour le rendre accessible aux lecteurs d'écran. Utilisation du composant TextArea

Vous pouvez utiliser un composant TextArea partout où vous avez besoin d'un champ de texte multiligne. Si vous avez besoin d'un champ de texte à ligne unique, utilisez le *Composant TextInput*, page 220. Par exemple, vous pouvez utiliser le composant TextArea comme un champ de commentaires dans un formulaire. Vous pouvez définir un écouteur qui vérifie si le champ est vide lorsqu'un utilisateur sort du champ. Cet écouteur peut afficher un message d'erreur indiquant qu'un commentaire doit être entré dans ce champ.

Paramètres du composant TextArea

Les paramètres suivants sont des paramètres de programmation que vous pouvez définir pour chaque occurrence de composant TextArea dans l'inspecteur des propriétés ou dans le panneau Inspecteur de composants :

text indique le contenu de TextArea. Vous ne pouvez pas entrer de retour chariot dans l'inspecteur des propriétés ou dans le panneau de l'Inspecteur de composants. La valeur par défaut est "" (chaîne vide).

html indique si le texte est formaté avec HTML (true) ou non (false). La valeur par défaut est false.

editable indique si le composant TextArea est modifiable (true) ou non (false). La valeur par défaut est true.

wordWrap indique si le texte est renvoyé à la ligne automatiquement (true) ou non (false). La valeur par défaut est true.

Vous pouvez rédiger du code ActionScript pour contrôler ces options ainsi que d'autres options des composants TextArea à l'aide des propriétés, méthodes et événements ActionScript. Pour plus d'informations, consultez Classe TextArea.

Création d'une application avec le composant TextArea

La procédure suivante explique comment ajouter un composant TextArea à une application en cours de programmation. Dans cet exemple, le composant est un champ Commentaire avec un écouteur d'événement qui détermine si l'utilisateur a entré du texte.

Pour créer une application avec le composant TextArea, effectuez les opérations suivantes :

1 Faites glisser un composant TextArea du panneau Composants jusqu'à la scène.

2 Dans l'inspecteur des propriétés, entrez **comment** comme nom d'occurrence.

3 Dans l'inspecteur des propriétés, définissez les paramètres de votre choix. Laissez toutefois le paramètre text vide, le paramètre editable défini sur true et le paramètre password sur false.

4 Choisissez l'image 1 dans le scénario, ouvrez le panneau Actions et saisissez le code suivant :

```
textListener = new Object();
textListener.handleEvent = function (evt){
  if (comment.length < 1)  {
  Alert(_root, "Erreur", "Vous devez entrer un commentaire dans ce champ",
  mxModal | mxOK);
  }
}
comment.addEventListener("FocusOut", textListener);
```

Ce code définit un gestionnaire d'événement `focusOut` sur l'occurrence `comment` de TextArea qui vérifie que l'utilisateur a bien tapé quelque chose dans le champ de texte.

5 Lorsque le texte est entré dans l'occurrence "comment", vous pouvez récupérer sa valeur comme suit :

```
var login = comment.text;
```

Personnalisation du composant TextArea

Vous pouvez transformer un composant TextArea horizontalement et verticalement, que ce soit en cours de programmation ou à l'exécution. Lors de la création, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez `UIObject.setSize()` ou toute propriété ou méthode applicable de la Classe TextArea.

Lorsqu'un composant TextArea est redimensionné, la bordure est redimensionnée en fonction du nouveau cadre de délimitation. Le cas échéant, les barres de défilement s'affichent sur les bords inférieur et droit du cadre. Le champ de texte est alors redimensionné dans la zone restante. Dans un composant TextArea, les éléments n'ont pas de taille fixe. Si le composant TextArea est trop petit pour afficher le texte, le texte est rogné.

Utilisation de styles avec le composant TextArea

Le composant TextArea prend en charge un jeu de styles de composant pour tout le texte du champ. Cependant, vous pouvez également afficher du code HTML compatible avec le rendu HTML de Flash Player. Pour afficher du texte HTML , définissez `TextArea.html` sur `true`.

Les propriétés de style `backgroundColor` et `borderStyle` du composant TextArea sont définies sur une déclaration de style de classe. Les styles de classe remplacent les styles _global. Pour définir les propriétés de style `backgroundColor` et `borderStyle`, vous devez donc créer une autre déclaration de style personnalisée sur l'occurrence.

Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, consultez *Utilisation des styles pour personnaliser la couleur et le texte des composants*, page 27.

Un composant TextArea prend en charge les styles suivants :

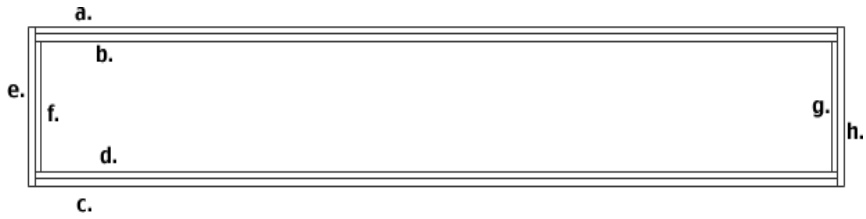
Style	Description
`color`	Couleur par défaut pour le texte.
`embedFonts`	Polices à incorporer dans le document.
`fontFamily`	Nom de police pour du texte.
`fontSize`	Taille en points pour la police.
`fontStyle`	Style de la police : "normal" ou "italic".
`fontWeight`	Epaisseur de la police : 'normal" ou "bold".
`textAlign`	Alignement du texte : "left", "right" ou "center".
`textDecoration`	Décoration du texte : "none" ou "underline".

Utilisation d'enveloppes avec le composant TextArea

Le composant TextArea utilise la classe RectBorder pour dessiner sa bordure. La méthode `setStyle()` (consultez `UIObject.setStyle()`) vous permet de modifier les propriétés suivantes du style RectBorder :

Styles RectBorder
`borderColor`
`highlightColor`
`borderColor`
`shadowColor`
`borderCapColor`
`shadowCapColor`
`shadowCapColor`
`borderCapColor`

Les propriétés de style définissent les positions suivantes sur la bordure :



Classe TextArea

Héritage UIObject > UIComponent > View > ScrollView > TextArea

Espace de nom de classe ActionScript mx.controls.TextArea

Les propriétés de la classe TextArea vous permettent de définir le contenu, le formatage et la position horizontale et verticale du texte à l'exécution. Vous pouvez également indiquer si le champ est modifiable et s'il s'agit d'un champ Mot de passe. Vous pouvez également limiter le nombre de caractères que l'utilisateur peut saisir.

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

Le composant TextArea annule le rectangle de focus par défaut de Flash Player et dessine un rectangle de focus personnalisé avec des angles arrondis.

Le composant TextArea supporte les styles CSS et tout style HTML supporté par Flash Player.

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

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

Remarque : Le code suivant renvoie la valeur undefined :
`trace(MonOccurrenceDeZoneDeTexte.version);`.

Propriétés de la classe TextArea

Propriété	Description
TextArea.editable	Une valeur booléenne indiquant si le champ est modifiable (true) ou non (false).
TextArea.hPosition	Définit la position du texte horizontalement dans le panneau défilant.
TextArea.hScrollPolicy	Indique si la barre de défilement horizontale est toujours affichée ("on"), jamais affichée ("off"), ou seulement lorsque nécessaire ("auto").S
TextArea.html	Un drapeau qui indique si le champ de texte peut être formaté avec HTML.
TextArea.length	Le nombre de caractères du champ de texte. Cette propriété est en lecture seule.
TextArea.maxChars	Le nombre maximum de caractères que le champ de texte peut contenir.
TextArea.maxHPosition	La valeur maximum de TextArea.hPosition.
TextArea.maxVPosition	La valeur maximum de TextArea.vPosition.
TextArea.password	Une valeur booléenne indiquant si le champ est un champ de mot de passe (true) ou non (false).
TextArea.restrict	Le jeu de caractères qu'un utilisateur peut entrer dans le champ de texte.
TextArea.text	Le contenu du texte d'un composant TextArea.
TextArea.vPosition	Un nombre indiquant la position du défilement vertical
TextArea.vScrollPolicy	Indique si la barre de défilement verticale est toujours affichée ("on"), jamais affichée ("off"), ou seulement lorsque nécessaire ("auto").S
TextArea.wordWrap	Une valeur booléenne indiquant si le texte est renvoyé à la ligne automatiquement (true) ou non (false).

Evénements de la classe TextArea

Evénement	Description
TextArea.change	Indique aux écouteurs que le texte a été modifié.

TextArea.change

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(change){
   ...
}
```

Usage 2 :

```
ObjetDécoute = new Object();
objetDécoute.change = function(objetEvnt){
   ...
}
OccurrenceDeZoneDeTexte.addEventListener("change", ObjetDécoute)
```

Description

Evénement : indique aux écouteurs que le texte a été modifié. Cet événement est diffusé après la modification du texte. Cet événement ne peut être utilisé pour empêcher l'ajout de certains caractères au champ de texte du composant. Utilisez plutôt `TextArea.restrict`.

Le premier exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence de composant TextArea. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de comportement. Par exemple, le code suivant, associé à l'occurrence `MaZoneDeTexte`, envoie « _level0.MaZoneDeTexte » au panneau de sortie :

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`OccurrenceDeZoneDeTexte`*) distribue un événement (ici, `change`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

Cet exemple comptabilise le nombre total de fois que le champ de texte a été modifié :

```
MaZoneDeTexte.changeHandler = function(obj) {
  this.changeCount++;
  trace(obj.target);
  trace("le texte a changé " + this.changeCount + " fois. Le champ contient
  désormais " +
this.text);
}
```

Consultez également

`UIEventDispatcher.addEventListener()`

TextArea.editable

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeZoneDeTexte.editable

Description

Propriété : valeur booléenne qui indique si le composant est modifiable (true) ou non (false). La valeur par défaut est true.

TextArea.hPosition

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeZoneDeTexte.hPosition

Description

Propriété : définit la position du texte horizontalement dans le champ. La valeur par défaut est 0.

Exemple

Le code suivant affiche les caractères les plus à gauche dans le champ :

```
MaZoneDeTexte.hPosition = 0;
```

TextArea.hScrollPolicy

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeZoneDeTexte.hScrollPolicy

Description

Propriété : détermine si la barre de défilement horizontale est toujours affichée ("on"), jamais affichée ("off") ou si elle doit s'afficher automatiquement en fonction de la taille du champ ("auto"). La valeur par défaut est "auto".

Exemple

Le code suivant permet un affichage systématique des barres de défilement horizontales :

```
text.hScrollPolicy = "on";
```

TextArea.html

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeZoneDeTexte.html

Description

Propriété : valeur booléenne indiquant si le champ de texte est formaté avec HTML (`true`) ou non (`false`). Si la propriété `html` est `true`, le champ de texte est un champ de texte html. Si `html` est `false`, le champ de texte n'est pas un champ de texte html. La valeur par défaut est `false`.

Exemple

L'exemple suivant transforme le champ `MaZoneDeTexte` en champ de texte HTML et formate le texte avec des balises HTML :

```
MaZoneDeTexte.html = true;
MaZoneDeTexte.text = "Le blabla <b>royal</b>"; // affiche "Le blabla royal"
```

TextArea.length

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeZoneDeTexte.length

Description

Propriété (lecture seule) : indique le nombre de caractères d'un champ de texte. Cette propriété renvoie la même valeur que la propriété `text.length` ActionScript, mais elle est plus rapide. Un caractère tel que tab (« \t ») compte comme un seul caractère. La valeur par défaut est 0.

Exemple

L'exemple suivant permet de récupérer la longueur du champ de texte et de la copier dans la variable `longueur` :

```
var longueur = MaZoneDeTexte.length; // trouve la longueur de la chaîne de
  texte
```

TextArea.maxChars

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeZoneDeTexte.maxChars

Description

Propriété : nombre maximum de caractères qu'un champ de texte peut contenir. Un script peut insérer plus de texte que la propriété `maxChars` ne le permet ; la propriété `maxChars` n'indique que la quantité de texte qu'un utilisateur peut entrer. Si la valeur de cette propriété est null, il n'y a pas de limite sur la quantité de texte qu'un utilisateur peut entrer. La valeur par défaut est null.

Exemple

L'exemple suivant permet de limiter à 255 le nombre de caractères qu'un utilisateur peut entrer :

```
MaZoneDeTexte.maxChars = 255;
```

TextArea.maxHPosition

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeZoneDeTexte.maxHPosition

Description

Propriété (lecture seule) : la valeur maximum de `TextArea.hPosition`. La valeur par défaut est 0.

Exemple

Le code suivant permet de faire défiler le texte complètement à droite :

```
MaZoneDeTexte.hPosition = MaZoneDeTexte.maxHPosition;
```

Consultez également

`TextArea.vPosition`

TextArea.maxVPosition

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeZoneDeTexte.maxVPosition

Description

Propriété (lecture seule) : indique la valeur maximum de TextArea.vPosition. La valeur par défaut est 0.

Exemple

Le code suivant permet de faire défiler le texte en bas du composant :

```
MaZoneDeTexte.vPosition = MaZoneDeTexte.maxVPosition;
```

Consultez également

TextArea.hPosition

TextArea.password

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeZoneDeTexte.password

Description

Propriété : valeur booléenne indiquant si le champ de texte est un champ de mot de passe (true) ou non (false). Si la valeur de password est true, le champ de texte est un champ de texte de mot de passe dont le contenu est masqué. Si false, le champ de texte n'est pas un champ de mode passe. La valeur par défaut est false.

Exemple

Le code suivant transforme le champ de texte en champ de mot de passe qui affiche tous les caractères sous la forme d'astérisques (*) :

```
MaZoneDeTexte.password = true;
```

TextArea.restrict

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeZoneDeTexte.restrict

Description

Propriété : indique le jeu de caractères qu'un utilisateur peut rentrer dans le champ de texte. La valeur par défaut est undefined. Si la valeur de la propriété restrict est null, un utilisateur peut entrer n'importe quel caractère. Si la valeur de la propriété `restrict` est une chaîne vide, aucun caractère ne peut être entré. Si la valeur de la propriété `restrict` est une chaîne de caractères, vous ne pouvez entrer que les caractères de la chaîne dans le champ de texte. La chaîne est lue de gauche à droite. Une plage peut être spécifiée en utilisant un tiret (-).

La propriété `restrict` limite seulement l'interaction avec l'utilisateur, un script pouvant mettre n'importe quel texte dans le champ de texte. Cette propriété ne se synchronise pas avec les cases à cocher de polices vectorielles intégrées de l'inspecteur des propriétés.

Si la chaîne commence par un caret « ^ », tous les caractères sont initialement acceptés et les caractères suivants de la chaîne sont exclus du jeu de caractères acceptés. Si la chaîne ne commence pas par un caret « ^ », aucun caractère n'est initialement accepté et les caractères suivants de la chaîne sont inclus dans le jeu de caractères acceptés.

Exemple

Dans l'exemple suivant, la première ligne de code limite le champ de texte aux lettres majuscules, aux nombres et aux espaces. La seconde ligne autorise tous les caractères, à l'exception des lettres minuscules.

```
mon_txt.restrict = "A-Z 0-9"; // limite le contrôle aux lettres majuscules, aux
  nombres et aux espaces
mon_txt.restrict = "^a-z"; // autorise tous les caractères, à l'exception des
  lettres minuscules
```

TextArea.text

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeZoneDeTexte.text

Description

Propriété : le contenu du texte d'un composant TextArea. La valeur par défaut est `""` (chaîne vide).

Exemple

Le code suivant permet de placer une chaîne dans l'occurrence `MaZoneDeTexte` et de présenter cette chaîne dans le panneau de sortie :

```
MaZoneDeTexte.text = "Le blabla royal";
trace(MaZoneDeTexte.text); // présente "Le blabla royal"
```

TextArea.vPosition

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`OccurrenceDeZoneDeTexte``.vPosition`

Description

Propriété : définit la position verticale du texte dans un champ de texte. La propriété scroll est utile pour diriger les utilisateurs vers un paragraphe spécifique dans un long passage, ou pour créer des champs de texte défilants. Vous pouvez obtenir et définir cette propriété. La valeur par défaut est 0.

Exemple

Le code suivant permet d'afficher les premiers caractères d'un champ :

```
MaZoneDeTexte.vPosition = 0;
```

TextArea.vScrollPolicy

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`OccurrenceDeZoneDeTexte``.vScrollPolicy`

Description

Propriété : détermine si la barre de défilement verticale est toujours affichée (`"on"`), jamais affichée (`"off"`), ou si elle s'affiche automatiquement en fonction de la taille du champ (`"auto"`). La valeur par défaut est `"auto"`.

Exemple

Le code suivant désactive systématiquement les barres de défilement verticales :

```
text.vScrollPolicy = "off";
```

TextArea.wordWrap

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeZoneDeTexte.wordWrap

Description

Propriété : valeur booléenne qui indique si le texte est renvoyé à la ligne automatiquement (`true`) ou non (`false`). La valeur par défaut est `true`.

Composant TextInput

TextInput est un composant à une seule ligne qui renvoie à la ligne automatiquement l'objet TextField ActionScript natif. Vous pouvez utiliser des styles pour personnaliser le composant TextInput. Lorsqu'une occurrence est désactivée, son contenu s'affiche dans une couleur représentée par le style "disabledColor". Un composant TextInput peut également être formaté avec HTML ou en tant que champ de mot de passe masquant le texte.

Un composant TextInput peut être activé ou désactivé dans une application. Lorsqu'il est désactivé, il ne reçoit pas les informations en provenance de la souris ou du clavier. Lorsqu'il est activé, il suit les mêmes règles de focus, de sélection et de navigation qu'un objet TextField ActionScript. Lorsqu'une occurrence TextInput a le focus, vous pouvez également utiliser les touches suivantes pour le contrôler :

Touche	Description
Touches de flèches	Déplace les caractères d'un caractère vers la gauche ou la droite.
Maj +Tab	Place le focus sur l'objet précédent.
Tab	Place le focus sur l'objet suivant.

Pour plus d'informations sur le contrôle du focus, consultez *Création de la navigation personnalisée du focus*, page 25 ou *Classe FocusManager*, page 103.

Un aperçu en direct de chaque occurrence TextInput permet de faire apparaître les modifications apportées dans les paramètres du panneau de l'inspecteur des propriétés ou des composants en cours de programmation. Lors d'un aperçu direct, il n'est pas possible de sélectionner du texte et vous ne pouvez pas entrer de texte dans l'occurrence du composant sur la scène.

Lorsque vous ajoutez un composant TextInput à une application, vous pouvez utiliser le panneau Accessibilité pour le rendre accessible aux lecteurs d'écran.

Utilisation du composant TextInput

Vous pouvez utiliser un composant TextInput là où vous avez besoin d'un champ de texte à une seule ligne. Si vous avez besoin d'un champ de texte multiligne, utilisez le *Composant TextArea*, page 207. Par exemple, vous pouvez utiliser un composant TextInput en tant que champ de mot de passe dans un formulaire. Vous pouvez définir un écouteur qui vérifie si le champ comporte suffisamment de caractères lorsque l'utilisateur sort du champ. Cet écouteur peut afficher un message d'erreur indiquant que l'utilisateur n'a pas entré le nombre de caractères adéquat.

Paramètres du composant TextInput

Les paramètres suivants sont des paramètres de programmation que vous pouvez définir pour chaque occurrence de composant TextInput dans le panneau de l'inspecteur des propriétés ou des composants :

text spécifie le contenu de TextInput. Vous ne pouvez pas entrer de retour chariot dans l'inspecteur des propriétés ou dans le panneau de l'Inspecteur de composants. La valeur par défaut est "" (chaîne vide).

editable indique si le composant TextInput est modifiable (true) ou non (false). La valeur par défaut est true.

password indique si le champ est un champ de mot de passe (true) ou non (false). La valeur par défaut est false.

Vous pouvez rédiger du code ActionScript pour contrôler ces options et d'autres options des composants TextInput en utilisant les propriétés, méthodes et événements ActionScript. Pour plus d'informations, consultez Classe TextInput.

Création d'une application avec le composant TextInput

La procédure suivante explique comment ajouter un composant TextInput à une application en cours de programmation. Dans cet exemple, le composant est un champ de mot de passe avec un écouteur d'événements qui détermine si l'utilisateur a entré le nombre de caractères adéquat.

Pour créer une application avec le composant TextInput, effectuez les opérations suivantes :

1. Faites glisser un composant TextInput du panneau Composants jusqu'à la scène.
2. Dans l'inspecteur des propriétés, entrez le nom d'occurrence **champMotDePasse**.
3. Dans l'inspecteur des propriétés, procédez comme suit :
   - Laissez le paramètre text vide.
   - Définissez le paramètre editable sur true.
   - Définissez le paramètre password sur true.
4. Choisissez l'image 1 dans le scénario, ouvrez le panneau Actions et saisissez le code suivant :

```
textListener = new Object();
textListener.handleEvent = function (evt){
  if (evt.type == "enter"){
    trace("Vous devez entrer au moins 8 caractères");
  }
}
champMotDePasse.addEventListener("enter", textListener);
```

Ce code définit un gestionnaire d'événement `enter` sur l'occurrence `champMotDePasse` de TextInput qui vérifie que l'utilisateur a entré le nombre de caractères adéquat.

5 Lorsque le texte est entré dans l'occurrence `champMotDePasse`, vous pouvez obtenir sa valeur comme suit :

```
var login = champMotDePasse.text;
```

Personnalisation du composant TextInput

Vous pouvez transformer un composant TextInput horizontalement en cours de programmation comme à l'exécution. Lors de la création, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez `UIObject.setSize()` ou toute propriété ou méthode applicable de la Classe TextInput.

Lorsqu'un composant TextInput est redimensionné, la bordure est prend la taille du nouveau cadre de délimitation. Le composant TextInput n'utilise pas de barres de défilement, mais le point d'insertion défile automatiquement lorsque l'utilisateur intervient sur le texte. Le champ de texte est alors redimensionné dans la zone restante. Les éléments d'un composant TextInput n'ont pas de taille fixe. Si le composant TextInput est trop petit pour afficher le texte, le texte est rogné.

Utilisation des styles avec le composant TextInput

Les propriétés de style `backgroundColor` et `borderStyle` du composant TextInput sont définies sur une déclaration de style de classe. Les styles de classe remplacent les styles _global. Pour définir les propriétés de style `backgroundColor` et `borderStyle`, vous devez donc créer une autre déclaration de style personnalisée sur l'occurrence.

Un composant TextInput prend en charge les styles suivants :

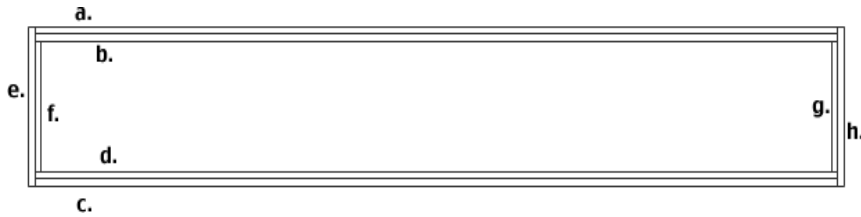
Style	Description
`color`	Couleur par défaut pour le texte.
`embedFonts`	Polices à incorporer dans le document.
`fontFamily`	Nom de police pour du texte.
`fontSize`	Taille en points pour la police.
`fontStyle`	Style de la police : "normal" ou "italic".
`fontWeight`	Epaisseur de la police : 'normal" ou "bold".
`textAlign`	Alignement du texte : "left", "right" ou "center".
`textDecoration`	Décoration du texte : "none" ou "underline".

Utilisation d'enveloppes avec le composant TextInput

Le composant TextArea utilise la classe RectBorder pour dessiner sa bordure. La méthode `setStyle()` (consultez `UIObject.setStyle()`) vous permet de modifier les propriétés suivantes du style RectBorder :

Styles RectBorder
`borderColor`
`highlightColor`
`borderColor`
`shadowColor`
`borderCapColor`
`shadowCapColor`
`shadowCapColor`
`borderCapColor`

Les propriétés de style définissent les positions suivantes sur la bordure :



Classe TextInput

Héritage UIObject > UIComponent > TextInput

Espace de nom de classe ActionScript mx.controls.TextInput

Les propriétés de la classe TextInput vous permettent de définir le contenu, le formatage et la position horizontale du texte à l'exécution. Vous pouvez également indiquer si le champ est modifiable et s'il s'agit d'un champ Mot de passe. Vous pouvez également limiter le nombre de caractères que l'utilisateur peut saisir.

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

Le composant TextInput utilise FocusManager pour annuler le rectangle de focus par défaut de Flash Player et pour dessiner un rectangle de focus personnalisé avec des angles arrondis. Pour plus d'informations, consultez *Classe FocusManager*, page 103.

Le composant TextInput supporte les styles CSS et tout style HTML supporté par Flash Player. Pour en savoir plus sur le support de CSS, consultez les spécifications du W3C.

Vous pouvez manipuler la chaîne de texte en utilisant la chaîne renvoyée par l'objet texte.

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

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

Remarque : Le code suivant renvoie la valeur undefined : `trace(monOccurrenceDeSaisieDeTexte.version);`.

Méthodes de la classe TextInput

Hérite de toutes les méthodes de la Classe UIObject et de la Classe UIComponent.

Propriétés de la classe TextInput

Propriété	Description
`TextInput.editable`	Une valeur booléenne indiquant si le champ est modifiable (`true`) ou non (`false`).
`TextInput.hPosition`	La position de défilement horizontale du champ de texte.
`TextInput.length`	Le nombre de caractères d'un champ de texte TextInput. Cette propriété est en lecture seule.
`TextInput.maxChars`	Le nombre maximum de caractères que l'utilisateur peut entrer dans un champ de texte TextInput.
`TextInput.maxHPosition`	La valeur maximum possible pour TextField.hPosition. Cette propriété est en lecture seule.
`TextInput.password`	Une valeur booléenne qui indique si le champ de saisie de texte est un champ de mot de passe qui masque les caractères saisis ou non.
`TextInput.restrict`	Indique les caractères que l'utilisateur peut entrer dans un champ de texte.
`TextInput.text`	Définit le contenu du texte d'un champ de texte TextInput.

Hérite de toutes les méthodes de la Classe UIObject et de la Classe UIComponent.

Evénements de la classe TextInput

Evénement	Description
`TextInput.change`	Déclenché lorsque le champ de saisie est modifié.
`TextInput.enter`	Déclenché lorsque l'utilisateur appuie sur la touche Entrée.

Hérite de toutes les méthodes de la Classe UIObject et de la Classe UIComponent.

TextInput.change

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(change){
   ...
}
```

Usage 2 :

```
ObjetDécoute = new Object();
objetDécoute.change = function(objetEvnt){
   ...
}
OccurrenceDeSaisieDeTexte.addEventListener("change", objetDécoute)
```

Description

Evénement : indique aux écouteurs que le texte a été modifié. Cet événement est diffusé après la modification du texte. Cet événement ne peut être utilisé pour empêcher l'ajout de certains caractères au champ de texte du composant. Utilisez plutôt `TextInput.restrict`. L'événement est déclenché uniquement par l'action de l'utilisateur, et non par une modification par programme.

Le premier exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence de composant TextInput. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de comportement. Par exemple, le code suivant, associé à l'occurrence `maSaisieDeTexte`, envoie « _level0.maSaisieDeTexte » au panneau de sortie :

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`OccurrenceDeSaisieDeTexte`*) distribue un événement (ici, `change`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

Cet exemple définit un drapeau dans l'application qui indique si le contenu du champ TextInput a été modifié :

```
form.change = function(objEvnt){
  // objEvnt.target est le composant ayant généré l'événement change,
  // par exemple, le composant Input.
  monFormulaireAChangé.visible = true;  // définit un indicateur de
  modification si le contenu a été modifié ;
}
maSaisie.addEventListener("change", form);
```

Consultez également

`UIEventDispatcher.addEventListener()`

TextInput.editable

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`OccurrenceDeSaisieDeTexte``.editable`

Description

Propriété : valeur booléenne qui indique si le composant est modifiable (`true`) ou non (`false`). La valeur par défaut est `true`.

TextInput.enter

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(enter){
  ...
}
```

Usage 2 :

```
ObjetDécoute = new Object();
objetDécoute.enter = function(objetEvnt){
  ...
}
OccurrenceDeSaisieDeTexte.addEventListener("enter", objetDécoute)
```

Description

Evénement : indique aux écouteurs que l'utilisateur a appuyé sur la touche Entrée.

Le premier exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence de composant TextInput. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de comportement. Par exemple, le code suivant, associé à l'occurrence `maSaisieDeTexte`, envoie « _level0.maSaisieDeTexte » au panneau de sortie :

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`OccurrenceDeSaisieDeTexte`*) distribue un événement (ici, `enter`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

Cet exemple définit un drapeau dans l'application qui indique si le contenu du champ TextInput a été modifié :

```
form.enter = function(eventObj){
  // eventObj.target est le composant qui a généré l'événement « enter »,
  // par exemple, le composant Input.
  monFormulaireAChangé.visible = true;
// définit un indicateur de modification si l'utilisateur appuie sur Entrée ;
}
maSaisie.addEventListener("enter", form);
```

Consultez également

`UIEventDispatcher.addEventListener()`

TextInput.hPosition

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`OccurrenceDeSaisieDeTexte``.hPosition`

Description

Propriété : définit la position du texte horizontalement dans le champ. La valeur par défaut est 0.

Exemple

Le code suivant affiche les caractères le plus à gauche dans le champ :

```
maSaisieDeTexte.hPosition = 0;
```

TextInput.length

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
OccurrenceDeSaisie.length
```

Description

Propriété (lecture seule) : un nombre qui indique le nombre de caractères d'un composant TextInput. Un caractère tel que tab (« \t ») compte comme un seul caractère. La valeur par défaut est 0.

Exemple

Le code suivant détermine le nombre de caractères de la chaîne `maSaisieDeTexte` et le copie dans la variable `length` :

```
var length = maSaisieDeTexte.length;
```

TextInput.maxChars

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
OccurrenceDeSaisieDeTexte.maxChars
```

Description

Propriété : nombre maximum de caractères qu'un champ de texte peut contenir. Un script peut insérer plus de texte que la propriété `maxChars` ne le permet ; la propriété `maxChars` n'indique que la quantité de texte qu'un utilisateur peut entrer. Si la valeur de cette propriété est null, il n'y a pas de limite sur la quantité de texte qu'un utilisateur peut entrer. La valeur par défaut est null.

Exemple

L'exemple suivant permet de limiter à 255 le nombre de caractères qu'un utilisateur peut entrer :

```
maSaisieDeTexte.maxChars = 255;
```

TextInput.maxHPosition

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeSaisieDeTexte.maxHPosition

Description

Propriété (lecture seule) : indique la valeur maximum de `TextInput.hPosition`. La valeur par défaut est 0.

Exemple

Le code suivant permet de faire défiler le texte complètement à droite :

```
maSaisieDeTexte.hPosition = maSaisieDeTexte.maxHPosition;
```

TextInput.password

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeSaisieDeTexte.password

Description

Propriété : valeur booléenne indiquant si le champ de texte est un champ de mot de passe (`true`) ou non (`false`). Si la valeur de `password` est `true`, le champ de texte est un champ de texte de mot de passe dont le contenu est masqué. Si `false`, le champ de texte n'est pas un champ de mode passe. La valeur par défaut est `false`.

Exemple

Le code suivant transforme le champ de texte en champ de mot de passe qui affiche tous les caractères sous la forme d'astérisques (*) :

```
maSaisieDeTexte.password = true;
```

TextInput.restrict

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

OccurrenceDeSaisieDeTexte.restrict

Description

Propriété : indique le jeu de caractères qu'un utilisateur peut rentrer dans le champ de texte. La valeur par défaut est undefined. Si la valeur de la propriété restrict est null ou une chaîne vide (""), l'utilisateur peut entrer n'importe quel caractère. Si la valeur de la propriété `restrict` est une chaîne de caractères, vous ne pouvez entrer que les caractères de la chaîne dans le champ de texte. La chaîne est lue de gauche à droite. Une plage peut être spécifiée en utilisant un tiret (-).

La propriété `restrict` limite seulement l'interaction avec l'utilisateur, un script pouvant mettre n'importe quel texte dans le champ de texte. Cette propriété ne se synchronise pas avec les cases à cocher de polices vectorielles intégrées de l'inspecteur des propriétés.

Si la chaîne commence par un caret « ^ », tous les caractères sont initialement acceptés et les caractères suivants de la chaîne sont exclus du jeu de caractères acceptés. Si la chaîne ne commence pas par un caret « ^ », aucun caractère n'est initialement accepté et les caractères suivants de la chaîne sont inclus dans le jeu de caractères acceptés.

La barre oblique inverse peut être utilisée pour entrer les caractères « - », « ^ » et « \ », comme ci-dessous.

```
\^
\-
\\
```

Le fait de placer une barre oblique inverse (\) entre guillemets ("") dans le panneau Actions a une signification particulière pour l'interpréteur de guillemets du panneau. Cette syntaxe indique en effet que le caractère suivant la barre oblique inverse (\) doit être traité comme tel. Le code suivant, par exemple, permet d'entrer une apostrophe :

```
var leftQuote = "\"";
```

L'interpréteur .restrict du panneau Actions utilise également la barre oblique inverse (\) comme caractère d'échappement. Vous penserez donc peut-être que la chaîne suivante est correcte :

```
monTexte.restrict = "0-9\-\^\\";
```

Toutefois, comme l'expression est placée entre guillemets, la valeur suivante est envoyée à l'interpréteur .restrict : `0-9-^\`. Cette valeur ne peut pas être interprétée.

L'interpréteur .restrict exigeant l'utilisation de guillemets, vous devez impérativement utiliser le caractère d'échappement pour que l'expression soit traitée correctement par l'interpréteur de guillemets intégré du panneau Actions. Pour envoyer la valeur `0-9\-\^\\` à l'interpréteur .restrict, vous devez donc entrer le code suivant :

```
monTexte.restrict = "0-9\\-\\^\\\\";
```

Exemple

Dans l'exemple suivant, la première ligne de code limite le champ de texte aux lettres majuscules, aux nombres et aux espaces. La seconde ligne autorise tous les caractères, à l'exception des lettres minuscules.

```
mon_txt.restrict = "A-Z 0-9";
mon_txt.restrict = "^a-z";
```

Le code suivant permet à un utilisateur d'entrer les caractères « 0 1 2 3 4 5 6 7 8 9 - ^ \ » dans l'occurrence `monTexte`. Vous devez utiliser une double barre oblique inverse pour échapper les caractères « -, ^, \ ». La première échappe les guillemets (“ ”), la seconde indique que le caractère suivant ne doit pas être traité comme un caractère spécial, comme illustré ci-dessous :

```
monTexte.restrict = "0-9\\-\\^\\\\";
```

TextInput.text

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
OccurrenceDeSaisieDeTexte.text
```

Description

Propriété : le contenu du texte d'un composant TextInput. La valeur par défaut est "" (chaîne vide).

Exemple

Le code suivant place une chaîne dans l'occurrence `maSaisieDeTexte`, puis présente cette chaîne au panneau de sortie :

```
maSaisieDeTexte.text = "Le blabla royal";
trace(maSaisieDeTexte.text); // présente "Le blabla royal"
```

Composant Tree

Pour obtenir les informations les plus récentes sur cette fonction, cliquez sur le bouton Mettre à jour dans la partie supérieure de l'onglet Aide.

Classe UIComponent

Héritage UIObject > UIComponent

Espace de nom de classe ActionScript mx.core.UIComponent

Tous les composants v2 étendent UIComponent. Il ne s'agit pas d'un composant visuel. La classe UIComponent contient des fonctions et des propriétés qui permettent aux composants Macromedia de partager un comportement commun. La classe UIComponent vous permet d'effectuer les opérations suivantes :

- Recevoir le focus et la saisie au clavier
- Activer et désactiver des composants
- Redimensionner en fonction de la disposition

Pour utiliser les méthodes et les propriétés de la classe UIComponent, appelez-la directement du composant que vous utilisez. Par exemple, pour appeler la méthode `UIComponent.setFocus()` du composant RadioButton, écrivez le code suivant :

```
monBoutonRadio.setFocus();
```

Pour créer un nouveau composant, vous avez uniquement besoin de créer une occurrence de la classe UIComponent si vous utilisez la version 2 des composants Macromedia. Même dans ce cas, la classe UIComponent est toujours créée de façon implicite par d'autres sous-classes telles que Button. Si vous n'avez pas besoin de créer une occurrence de la classe UIComponent, utilisez le code suivant :

```
class MonComposant extends UIComponent;
```

Méthodes de la classe UIComponent

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

Hérite de toutes les méthodes de la classe Classe UIObject.

Propriétés de la classe UIComponent

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

Hérite de toutes les propriétés de la classe Classe UIObject.

Evénements de la classe UIComponent

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

Hérite de tous les événements de la classe Classe UIObject.

UIComponent.focusIn

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
on(focusIn){
  ...
}
ObjetDécoute = new Object();
```

```
objetDécoute.focusIn = function(objetEvnt){
  ...
}
occurrenceDeComposant.addEventListener("focusIn", objetDécoute)
```

Description

Evénement : indique aux écouteurs que l'objet a reçu le focus clavier.

Le premier exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence de composant.

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`occurrenceDeComposant`*) distribue un événement (ici, `focusIn`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

Le code suivant désactive un bouton lorsque l'utilisateur renseigne le champ de texte `txt` :

```
txtListener.handleEvent = function(objetEvnt){
  form.button.enabled = false;
}
txt.addEventListener("focusIn", txtListener);
```

Consultez également

`UIEventDispatcher.addEventListener()`

UIComponent.focusOut

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
on(focusOut){
  ...
}
ObjetDécoute = new Object();
objetDécoute.focusOut = function(objetEvnt){
  ...
}
occurrenceDeComposant.addEventListener("focusOut", objetDécoute)
```

Description

Evénement : indique aux écouteurs que l'objet a perdu le focus clavier.

Le premier exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence de composant.

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`occurrenceDeComposant`*) distribue un événement (ici, focusOut) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

Le code suivant active un bouton lorsque l'utilisateur quitte le champ de texte `txt` :

```
txtListener.handleEvent = function(objetEvnt){
  if (objetEvnt.type == focusOut){
      form.button.enabled = true;
  }
}
txt.addEventListener("focusOut", txtListener);
```

Consultez également

`UIEventDispatcher.addEventListener()`

UIComponent.enabled

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`occurrenceDeComposant``.enabled`

Description

Propriété : indique si le composant peut accepter le focus et les actions de la souris. Si la valeur est `true`, il peut recevoir le focus et les actions de la souris ; si la valeur est `false`, il ne peut pas. La valeur par défaut est `true`.

Exemple

L'exemple suivant définit la propriété `enabled` d'un composant CheckBox sur `false` :

```
occurrenceDeCaseACocher.enabled = false;
```

UIComponent.getFocus()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
occurrenceDeComposant.getFocus();
```

Paramètres

Aucun.

Renvoie

Une référence à l'objet qui a actuellement le focus.

Description

Méthode : renvoie une référence à l'objet qui a le focus clavier.

Exemple

Le code suivant renvoie une référence à l'objet qui a le focus et l'affecte à la variable `tmp` :

```
var tmp = checkbox.getFocus();
```

UIComponent.keyDown

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
on(keyDown){
  ...
}
ObjetDécoute = new Object();
objetDécoute.keyDown = function(objetEvnt){
  ...
}
occurrenceDeComposant.addEventListener("keyDown", objetDécoute)
```

Description

Evénement : indique aux écouteurs lorsque l'utilisateur appuie sur une touche. Il s'agit d'un événement de très bas niveau qui doit être utilisé uniquement en cas de nécessité, car il pourrait avoir une incidence sur les performances du système.

Le premier exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence de composant.

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*occurrenceDeComposant*) distribue un événement (ici, keyDown)) qui est géré par une fonction associée à un objet d'écoute (*objetDécoute*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*objetEvnt*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

Le code suivant fait clignoter une icône lorsque l'utilisateur appuie sur une touche :

```
formListener.handleEvent = function(objEvnt)
{
  form.icon.visible = !form.icon.visible;
}
form.addEventListener("keyDown", formListener);
```

UIComponent.keyUp

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
on(keyUp){
  ...
}
ObjetDécoute = new Object();
objetDécoute.keyUp = function(objetEvnt){
  ...
}
occurrenceDeComposant.addEventListener("keyUp", objetDécoute)
```

Description

Evénement : signale aux écouteurs que l'utilisateur a relâché une touche. Il s'agit d'un événement de très bas niveau qui doit être utilisé uniquement en cas de nécessité, car il pourrait avoir une incidence sur les performances du système.

Le premier exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence de composant.

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*occurrenceDeComposant*) distribue un événement (ici, keyUp) qui est géré par une fonction associée à un objet d'écoute (*objetDécoute*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*objetEvnt*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

Le code suivant fait clignoter une icône lorsqu'une touche est relâchée :

```
formListener.handleEvent = function(objEvnt)
{
  form.icon.visible = !form.icon.visible;
}
form.addEventListener("keyUp", formListener);
```

UIComponent.setFocus()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
occurrenceDeComposant.setFocus();
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : définit le focus à cette occurrence de composant. L'occurrence avec le focus reçoit la saisie au clavier.

Exemple

Le code suivant définit le focus à l'occurrence `checkbox` :

```
checkbox.setFocus();
```

UIComponent.tabIndex

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
instance.tabIndex
```

Description

Propriété : nombre indiquant l'ordre de tabulation pour un composant dans un document.

Exemple

Le code suivant donne la valeur `tmp` à la propriété `tabIndex` de l'occurrence `checkbox` :

```
var tmp = checkbox.tabIndex;
```

Classe UIEventDispatcher

Espace de nom de classe ActionScript mx.events.EventDispatcher; mx.events.UIEventDispatcher

Les événements vous permettent de savoir quand l'utilisateur a interagi avec un composant. Ils vous avertissent également lorsque des modifications importantes se produisent dans l'apparence ou le cycle de vie d'un composant, telles que sa création, sa destruction ou son redimensionnement.

Chaque composant diffuse différents événements et ces événements sont répertoriés dans chaque entrée de composant. Les événements dans le code ActionScript peuvent s'utiliser de différentes façons. Pour plus d'informations, consultez *A propos des événements de composant*, page 22.

Utilisez `UIEventDispatcher.addEventListener()` pour enregistrer un écouteur avec une occurrence de composant. L'écouteur est invoqué lorsque l'événement d'un composant est déclenché.

UIEventDispatcher.addEventListener()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004 et Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.addEventListener(événement, écouteur)
```

Paramètres

événement Chaîne portant le même nom que l'événement.

écouteur Référence à une fonction ou à un objet d'écoute.

Renvoie

Rien.

Description

Méthode : enregistre un objet d'écoute avec une occurrence de composant qui diffuse un événement. Lorsque l'événement est déclenché, l'objet d'écoute ou fonction est averti(e). Vous pouvez appeler cette méthode à partir de toute occurrence de composant. Le code suivant, par exemple, enregistre un écouteur dans l'occurrence de composant `monBouton` :

```
monBouton.addEventListener("click", monEcouteur);
```

Vous devez définir l'écouteur en tant qu'objet ou fonction avant d'appeler `addEventListener()` pour enregistrer l'écouteur dans l'occurrence de composant. Si l'écouteur est un objet, il doit avoir une fonction de rappel, invoquée lorsque l'événement est déclenché. En général, cette fonction de rappel porte le même nom que l'événement avec lequel l'écouteur est enregistré. Si l'écouteur est une fonction, la fonction est invoquée lorsque l'événement est déclenché. Pour plus d'informations, consultez *Utilisation des écouteurs d'événements de composant*, page 23.

Vous pouvez enregistrer plusieurs écouteurs dans une même occurrence de composant, mais vous devez appeler `addEventListener()` séparément pour chacun d'eux. De même, vous pouvez enregistrer un écouteur dans plusieurs occurrences de composant, mais vous devez appeler `addEventListener()` séparément pour chacune d'elles. Le code suivant, par exemple, définit un objet d'écoute et l'affecte à deux occurrences de composant Button :

```
lo = new Object();
lo.click = function(evt){
  if (evt.target == button1){
    trace("On a cliqué sur le bouton 1");
  } else if (evt.target == button2){
    trace("On a cliqué sur le bouton 2");
  }
}
button1.addEventListener("click", lo);
button2.addEventListener("click", lo);
```

Un objet événement est transmis à l'écouteur comme un paramètre. L'objet événement a des propriétés contenant des informations sur l'événement qui s'est produit. Vous pouvez utiliser l'objet événement à l'intérieur de la fonction de rappel de l'écouteur pour déterminer le type d'événement qui s'est produit et l'occurrence qui a diffusé l'événement. Dans l'exemple ci-dessus, l'objet événement correspond à `evt` (vous pouvez utiliser n'importe quel identificateur comme nom d'objet événement) et il est utilisé dans les instructions `if` pour déterminer sur quelle occurrence de bouton l'utilisateur a cliqué. Pour plus d'informations, consultez *Objets événement*, page 240.

Exemple

L'exemple suivant définit un objet d'écoute, `monEcouteur` et définit le clic de la fonction de rappel. Il appelle ensuite `addEventListener()` pour enregistrer l'objet d'écoute `monEcouteur` avec l'occurrence de composant `monBouton`. Pour tester ce code, placez sur la scène un composant Button portant le nom d'occurrence `monBouton`, et insérez le code suivant dans l'image 1 :

```
monEcouteur = new Object();
monEcouteur.click = function(evt){
  trace(evt.type + " triggered");
}
monBouton.addEventListener("click", monEcouteur);
```

Objets événement

Un objet événement est transmis à un écouteur en tant que paramètre. L'objet événement est un objet ActionScript dont les propriétés contiennent des informations sur l'événement qui s'est produit. Vous pouvez l'utiliser à l'intérieur de la fonction de rappel de l'écouteur pour déterminer le nom de l'événement diffusé ou le nom d'occurrence du composant qui a diffusé l'événement. Le code suivant, par exemple, utilise la propriété `target` de l'objet événement `objEvt` pour accéder à la propriété `label` de l'occurrence `monBouton` et envoyer la valeur au panneau de sortie :

```
listener = new Object();
listener.click = function(objEvt){
  trace("On a cliqué sur le bouton " + objEvt.target.label);
}
monBouton.addEventListener("click", listener);
```

Certaines propriétés d'objet événement sont définies dans les spécifications du W3C, mais ne sont pas implémentées dans la version 2 (v2) de Macromedia Component Architecture. Tous les objets événement v2 possèdent les propriétés énumérées dans le tableau ci-dessous. Des propriétés supplémentaires sont définies pour certains événements. Dans ce cas, elles sont indiquées dans l'entrée correspondant à l'événement.

Propriétés de l'objet événement

Propriété	Description
type	Chaîne indiquant le nom de l'événement.
cible	Référence à l'occurrence de composant qui émet l'événement.

Classe UIObject

Héritage MovieClip > UIObject

Espace de nom de classe ActionScript mx.core.UIObject

UIObject est la classe de base pour tous les composants v2. Il ne s'agit pas d'un composant visuel. La classe UIObject enveloppe l'objet MovieClip ActionScript et contient des fonctions et des propriétés qui permettent aux composants Macromedia v2 de partager des comportements communs. La classe UIObject implémente les éléments suivants :

- Styles
- Evénements
- Redimensionnement par mise à l'échelle

Pour utiliser les méthodes et propriétés de la classe UIObject, appelez-les directement du composant que vous utilisez. Par exemple, pour appeler la méthode `UIObject.setSize()` du composant RadioButton, écrivez le code suivant :

```
monBoutonRadio.setSize(30, 30);
```

Vous n'avez besoin de créer une occurrence de la classe UIObject que si vous utilisez Macromedia Component V2 Architecture pour créer un nouveau composant. Même dans ce cas, UIObject est souvent créée de façon implicite par d'autres classes, telles que Button. Si vous n'avez pas besoin de créer une occurrence de classe UIObject, utilisez le code suivant :

```
class MonComposant extends UIObject;
```

Méthodes de la classe UIObject

Méthode	Description
`UIObject.createObject()`	Crée un sous-objet sur un objet.
`UIObject.createClassObject()`	Crée un objet sur la classe spécifiée.
`UIObject.destroyObject()`	Détruit une occurrence de composant.
`UIObject.invalidate()`	Marque l'objet de sorte qu'il soit redessiné sur le prochain intervalle d'image.
`UIObject.move()`	Déplace l'objet à l'emplacement demandé.
`UIObject.redraw()`	Force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.
`UIObject.setSize()`	Redimensionne l'objet aux dimensions demandées.
`UIObject.setSkin()`	Définit une enveloppe dans l'objet.

Propriétés de la classe UIObject

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

Evénements de la classe UIObject

Evénement	Description
`UIObject.draw`	Diffusion lorsqu'un objet est sur le point de dessiner ses graphiques.
`UIObject.load`	Diffusion lorsque des sous-objets sont créés.
`UIObject.move`	Diffusion lorsque l'objet a été déplacé.
`UIObject.resize`	Diffusion lorsque les sous-objets sont vidés.
`UIObject.unload`	Diffusion lorsque les sous-objets sont vidés.

UIObject.bottom

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.bottom

Description

Propriété (lecture seule) : nombre indiquant la position inférieure de l'objet, en pixels, par rapport à celle de son parent.

Exemple

Définit la valeur tmp à la position inférieure de la case à cocher :

```
var tmp = checkbox.bottom;
```

UIObject.createObject()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.createObject(*nomLiaison*, *nomDoccurrence*, *profondeur*, *objetInit*)

Paramètres

nomLiaison Une chaîne indiquant l'identificateur de liaison d'un symbole dans le panneau Bibliothèque.

nomDoccurrence Une chaîne indiquant le nom de l'occurrence de la nouvelle occurrence.

profondeur Un nombre indiquant la profondeur de la nouvelle occurrence.

objetInit Un objet contenant des propriétés d'initialisation pour la nouvelle occurrence.

Renvoie

Une classe UIObject qui est une occurrence du symbole.

Description

Méthode : crée un sous-objet sur un objet. Utilisée, en général, uniquement par les développeurs de composants ou les développeurs confirmés.

Exemple

L'exemple suivant crée une occurrence CheckBox sur l'objet `form` :

```
form.createObject("CheckBox", "sym1", 0);
```

UIObject.createClassObject()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.createClassObject(*nomClasse*, *nomDoccurrence*, *profondeur*, *objetInit*)

Paramètres

nomClasse Un objet indiquant la classe de la nouvelle occurrence.

nomDoccurrence Une chaîne indiquant le nom de l'occurrence de la nouvelle occurrence.

profondeur Un nombre indiquant la profondeur de la nouvelle occurrence.

objetInit Un objet contenant des propriétés d'initialisation pour la nouvelle occurrence.

Renvoie

Une classe UIObject qui est une occurrence de la classe spécifiée.

Description

Méthode : crée un sous-objet d'un objet. Utilisée, en général, uniquement par les développeurs de composants ou les développeurs confirmés. Cette méthode vous permet de créer des composants à l'exécution.

Vous devez spécifier le nom de paquet de la classe. Procédez de l'une des manières suivantes :

```
import mx.controls.Button;
createClassObject(Button,"button2",5,{label:"Test Button"});
```

ou

```
createClassObject(mx.controls.Button,"button2",5,{label:"Test Button"});
```

Exemple

L'exemple suivant crée un objet CheckBox :

```
form.createClassObject(CheckBox, "cb", 0, {label:"Check this"});
```

UIObject.destroyObject()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
occurrenceDeComposant.destroyObject(nomOccurrence)
```

Paramètres

nomOccurrence Chaîne indiquant le nom d'occurrence de l'objet à détruire.

Renvoie

Rien.

Description

Méthode : détruit une occurrence de composant.

UIObject.draw

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
on(draw){
  ...
}
ObjetDécoute = new Object();
objetDécoute.draw = function(objetEvnt){
  ...
}
occurrenceDeComposant.addEventListener("draw", objetDécoute)
```

Description

Evénement : indique aux écouteurs que l'objet est sur le point de dessiner ses graphiques. Il s'agit d'un événement de très bas niveau qui doit être utilisé uniquement en cas de nécessité, car il pourrait avoir une incidence sur les performances du système.

Le premier exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence de composant.

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*occurrenceDeComposant*) distribue un événement (ici, draw) qui est géré par une fonction associée à un objet d'écoute (*objetDécoute*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*objetEvnt*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

Le code suivant redessine l'objet `form2` lorsque l'objet `form` est dessiné :

```
formListener.draw = function(objEvt){
form2.redraw(true);
}
form.addEventListener("draw", formListener);
```

Consultez également

`UIEventDispatcher.addEventListener()`

UIObject.height

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.height

Description

Propriété (lecture seule) : un nombre indiquant la hauteur de l'objet, en pixels.

Exemple

L'exemple suivant définit la valeur `tmp` à la hauteur de l'occurrence `checkbox` :

```
var tmp = checkbox.height;
```

UIObject.getStyle()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.getStyle(*nomDeProp*)

Paramètres

nomDeProp Une chaîne indiquant le nom de la propriété de style (par exemple, "fontWeight", "borderStyle", etc.).

Renvoie

La valeur de la propriété de style. La valeur peut être de n'importe quel type de données.

Description

Méthode : récupère la propriété de style à partir de styleDeclaration ou de l'objet. Si la propriété de style est un style d'héritage, les parents de l'objet peuvent être la source de la valeur du style.

Pour obtenir une liste des styles supportés par chaque composant, consultez les entrées correspondantes.

Exemple

Le code suivant définit la propriété de style fontWeight de l'occurrence ib sur bold si la propriété de style fontWeight de l'occurrence cb est bold :

```
if (cb.getStyle("fontWeight") == "bold")
{
   ib.setStyle("fontWeight", "bold");
};
```

UIObject.invalidate()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.invalidate()

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : marque l'objet pour qu'il soit redessiné sur le prochain intervalle d'image.

Exemple

L'exemple suivant marque pBar de l'occurrence ProgressBar pour que l'objet soit redessiné :

```
pBar.invalidate();
```

UIObject.left

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
occurrenceDeComposant.left
```

Description

Propriété (lecture seule) : un nombre indiquant le bord gauche de l'objet, en pixels.

Exemple

L'exemple suivant donne à la valeur `tmp` la position gauche de l'occurrence `checkbox` :

```
var tmp = checkbox.left; // donne à la valeur tmp la position gauche de checkbox ;
```

UIObject.load

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(load){
   ...
}
```

Usage 2 :

```
ObjetDécoute = new Object();
objetDécoute.load = function(objetEvnt){
   ...
}
occurrenceDeComposant.addEventListener("load", objetDécoute)
```

Description

Evénement : indique aux écouteurs que le sous-objet pour cet objet est en cours de création.

Le premier exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence de composant.

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*occurrenceDeComposant*) distribue un événement (ici, load) qui est géré par une fonction associée à un objet d'écoute (*objetDécoute*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*objetEvnt*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

L'exemple suivant crée une occurrence de MonSymbole une fois que l'occurrence `form` est chargée :

```
formListener.handleEvent = function(objEvnt)
{
  form.createObject("MonSymbole", "sym1", 0);
}
form.addEventListener("load", formListener);
```

UIObject.move

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(move){
  ...
}
```

Usage 2 :

```
ObjetDécoute = new Object();
objetDécoute.move = function(objetEvnt){
  ...
}
occurrenceDeComposant.addEventListener("move", objetDécoute)
```

Description

Evénement : indique aux écouteurs que l'objet a été déplacé.

Le premier exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence de composant.

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*occurrenceDeComposant*) distribue un événement (ici, move) qui est géré par une fonction associée à un objet d'écoute (*objetDécoute*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*objetEvnt*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

L'exemple suivant appelle la méthode `move()` pour conserver `form2` à 100 pixels vers le bas, à droite de `form1` :

```
formListener.handleEvent = function(objEvnt)
{
  // objEvnt.target est le composant qui a généré l'événement change
  form2.move(form1.x + 100, form1.y + 100);
}
form1.addEventListener("move", formListener);
```

UIObject.move()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.move(*x*, *y*)

Paramètres

x Un nombre qui indique la position du coin supérieur gauche de l'objet par rapport à son parent.

y Un nombre qui indique la position du coin supérieur droit de l'objet par rapport à son parent.

Renvoie

Rien.

Description

Méthode : déplace l'objet à l'emplacement demandé. Il est recommandé de transmettre uniquement des valeurs entières à la fonction `UIObject.move()`, sinon le composant risque d'apparaître flou.

Le non-respect de ces règles peut générer des commandes encore plus floues.

Exemple

Cet exemple déplace `pBar` de l'occurrence ProgressBar dans le coin supérieur gauche à 100, 100 :

```
pBar.move(100, 100);
```

UIObject.redraw()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`occurrenceDeComposant``.redraw()`

Paramètres

`toujours` Une valeur booléenne `true` pour toujours redessiner, `false` pour redessiner uniquement en cas d'annulation.

Renvoie

Rien.

Description

Méthode : force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle

Exemple

Cet exemple force l'occurrence `pBar` à redessiner immédiatement :

```
pBar.validate(true);
```

UIObject.resize

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(resize){
  ...
}
```

Usage 2 :

```
ObjetDécoute = new Object();
objetDécoute.resize = function(objetEvnt){
  ...
}
occurrenceDeComposant.addEventListener("resize", objetDécoute)
```

Description

Evénement : indique aux écouteurs que les sous-objets de cet objet sont vidés.

Le premier exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence de composant.

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`occurrenceDeComposant`*) distribue un événement (ici, resize) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

L'exemple suivant appelle la méthode `setSize()` pour que `sym1` fasse la moitié de la largeur et un quart de la hauteur lorsque `form` est déplacé :

```
formListener.handleEvent = function(objEvnt)
  form.sym1.setSize(sym1.width / 2, sym1.height / 4);
}
form.addEventListener("resize", formListener);
```

UIObject.right

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`occurrenceDeComposant``.right`

Description

Propriété (lecture seule) : un nombre indiquant la position de l'objet sur la droite, en pixels, par rapport au côté droit de son parent.

Exemple

L'exemple suivant définit la valeur `tmp` à la position droite de l'occurrence `checkbox` :

```
var tmp = checkbox.right;
```

UIObject.scaleX

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.scaleX

Description

Propriété : nombre indiquant le facteur de redimensionnement dans la direction *x* de l'objet par rapport à son parent.

Exemple

L'exemple suivant double la largeur de la case à cocher et définit la variable `tmp` par rapport au facteur de redimensionnement horizontal :

```
checkbox.scaleX = 200;
var tmp = checkbox.scaleX;
```

UIObject.scaleY

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.scaleY

Description

Propriété : nombre indiquant le facteur de redimensionnement dans la direction y de l'objet par rapport à son parent.

Exemple

L'exemple suivant double la hauteur de la case à cocher et définit la variable `tmp` par rapport au facteur de redimensionnement vertical :

```
checkbox.scaleY = 200;
var tmp = checkbox.scaleY;
```

UIObject.setSize()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.setSize(*largeur*, *hauteur*)

Paramètres

largeur Nombre indiquant la largeur de l'objet en pixels.

hauteur Nombre indiquant la hauteur de l'objet en pixels.

Renvoie

Rien.

Description

Méthode : redimensionne l'objet à la taille requise. Il est recommandé de transmettre uniquement des valeurs entières à la fonction `UIObject.setSize()`, sinon le composant risque d'apparaître flou. Cette méthode (ainsi que toutes les méthodes et propriétés de la classe UIObject) est disponible à partir de n'importe quelle occurrence de composant.

Lorsque vous appelez cette méthode sur une occurrence de ComboBox, la liste déroulante est redimensionnée et la propriété `rowHeight` de la liste contenue est également modifiée.

Exemple

L'exemple suivant fixe la taille du composant `pBar` à 100 pixels de largeur et à 100 pixels de hauteur :

```
pBar.setSize(100, 100);
```

UIObject.setSkin()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.setSkin(*id*, *nomLiaison*)

Paramètres

id Un nombre correspondant à la variable. Cette valeur est en général une constante définie dans la définition de classe.

nomLiaison Une chaîne indiquant un actif de la bibliothèque.

Renvoie

Rien.

Description

Méthode : définit une enveloppe dans l'occurrence de composant. Servez-vous de cette méthode lorsque vous programmez des composants. Elle ne permet pas de définir les enveloppes d'un composant lors de l'exécution.

Exemple

Cet exemple définit une enveloppe dans l'occurrence `checkbox` :

```
checkbox.setSkin(CheckBox.skinIDCheckMark, "MaCochePersonnalisée");
```

UIObject.setStyle()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
occurrenceDeComposant.setStyle(nomDeProp, valeur)
```

Paramètres

nomDeProp Une chaîne indiquant le nom de la propriété de style (par exemple, `"fontWeight"`, `"borderStyle"`, etc.).

valeur La valeur de la propriété.

Renvoie

Une classe UIObject qui est une occurrence de la classe spécifiée.

Description

Méthode : définit la propriété de style sur l'objet ou la déclaration de style. Si la propriété de style est un style d'héritage, les enfants de l'objet sont informés de la nouvelle valeur.

Pour obtenir une liste des styles supportés par chaque composant, consultez les entrées correspondantes.

Exemple

Le code suivant définit la propriété de style `fontWeight` de l'occurrence de la case à cocher `cb` sur bold :

```
cb.setStyle("fontWeight", "bold");
```

UIObject.top

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
occurrenceDeComposant.top
```

Description

Propriété (lecture-seule) : un nombre indiquant le bord supérieur de l'objet, en pixels.

Exemple

L'exemple suivant définit la variable `tmp` à la position supérieure de l'occurrence `checkbox` :

```
var tmp = checkbox.top; // définit la valeur de tmp à la position supérieure de
  la case à cocher ;
```

UIObject.unload

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(unload){
   ...
}
```

Usage 2 :

```
ObjetDécoute = new Object();
objetDécoute.unload = function(objetEvnt){
   ...
}
occurrenceDeComposant.addEventListener("unload", objetDécoute)
```

Description

Evénement : indique aux écouteurs que les sous-objets de cet objet sont vidés.

Le premier exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence de composant.

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`occurrenceDeComposant`*) distribue un événement (ici, unload) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. Chaque objet événement dispose d'un jeu de propriétés contenant des informations relatives à l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

L'exemple suivant supprime `sym1` lorsque l'événement unload est déclenché :

```
formListener.handleEvent = function(objEvnt)
  // objEvnt.target est le composant ayant généré l'événement change,
  form.destroyObject(sym1);
```

```
}
form.addEventListener("unload", formListener);
```

UIObject.visible

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.visible

Description

Propriété : valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).

Exemple

L'exemple suivant rend visible l'occurrence du composant Loader `monChargeur` :

```
monChargeur.visible = true;
```

UIObject.width

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.width

Description

Propriété (lecture seule) : un nombre indiquant la largeur de l'objet, en pixels.

Exemple

L'exemple suivant fixe à 450 pixels la largeur du composant TextArea :

```
maZoneDeTexte.width = 450;
```

UIObject.x

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.x

Description

Propriété (lecture seule) : un nombre indiquant le bord gauche de l'objet, en pixels.

Exemple

L'exemple suivant fixe à 150 le bord gauche de la case à cocher :

```
caseACocher.x = 150;
```

UIObject.y

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

```
occurrenceDeComposant.y
```

Description

Propriété (lecture seule) : un nombre indiquant le bord supérieur de l'objet, en pixels.

Exemple

L'exemple suivant fixe à 200 le bord supérieur de la case à cocher :

```
caseACocher.y = 200;
```

Paquet WebServices

Pour des informations actualisées sur cette fonction, cliquez sur le bouton Mettre à jour dans la partie supérieure de l'onglet Aide.

Composant WebServiceConnector

Pour des informations actualisées sur cette fonction, cliquez sur le bouton Mettre à jour dans la partie supérieure de l'onglet Aide.

Composant Window

Un composant Window affiche le contenu d'un clip dans une fenêtre dotée d'une barre de titre, d'une bordure et d'un bouton Fermer (en option).

Un composant Window peut être modal ou non modal. Une fenêtre modale empêche les actions de la souris ou la saisie au clavier dans des composants qui se trouvent à l'extérieur de la fenêtre. Le composant Window peut également être déplacé. L'utilisateur peut cliquer sur la barre de titre et faire glisser la fenêtre et son contenu à un autre emplacement. Le fait de faire glisser les bordures ne redimensionne pas la fenêtre.

Si vous utilisez PopUpManager pour insérer un composant Window dans un document, l'occurrence Window possède son propre gestionnaire de focus (FocusManager), qui est distinct du reste du document. Si vous n'utilisez pas PopUpManager, le contenu de la fenêtre participe à l'ordre des focus. Pour plus d'informations sur le contrôle du focus, consultez *Création de la navigation personnalisée du focus*, page 25 ou *Classe FocusManager*, page 103.

Un aperçu en direct de chaque occurrence Window permet de faire apparaître, au cours de la programmation, les modifications qui ont été apportées à tous les paramètres, à l'exception de contentPath, dans l'inspecteur des propriétés ou le panneau Composants.

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

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

Vous n'activez l'accessibilité d'un composant qu'une fois, quel que soit son nombre d'occurrences. Pour plus d'informations, consultez « Création de contenu accessible », dans le guide Utilisation de Flash de l'aide. Il vous faudra parfois mettre votre système d'aide à jour pour obtenir ces informations.

Utilisation du composant Window

Vous pouvez utiliser une fenêtre dans une application dès que vous avez besoin de présenter à l'utilisateur une information ou un choix prioritaire par rapport au reste de l'application. Par exemple, si vous souhaitez que l'utilisateur renseigne une fenêtre de connexion ou une fenêtre qui permet de modifier un mot de passe et d'en confirmer un nouveau.

Il existe plusieurs manières d'ajouter une fenêtre à une application. Vous pouvez faire glisser une fenêtre du panneau Composants jusqu'à la scène. Vous pouvez aussi utiliser un appel `createClassObject()` (consultez `UIObject.createClassObject()`) pour ajouter une fenêtre à une application. La troisième façon d'ajouter une fenêtre à une application est d'utiliser Classe PopUpManager. Utilisez PopUpManager pour créer des fenêtres modales qui se superposent aux autres objets de la scène. Pour plus d'informations, consultez Classe Window.

Paramètres du composant Window

Les paramètres suivants sont des paramètres de programmation que vous pouvez définir pour chaque occurrence de composant Window dans le panneau de l'inspecteur des propriétés ou des composants :

contentPath spécifie le contenu de la fenêtre. Il peut s'agir de l'identificateur de liaison du clip ou du nom de symbole d'un écran, d'un formulaire ou d'une diapositive qui contient le contenu de la fenêtre. Il peut également s'agir d'une URL absolue ou relative pour un fichier SWF ou JPG à charger dans la fenêtre. La valeur par défaut est "". Le contenu chargé est coupé pour être adapté à la fenêtre.

title indique le titre de la fenêtre.

closeButton indique si un bouton Fermer est affiché (true) ou non (false). Cliquer sur le bouton de fermeture diffuse un événement `click`, mais ne ferme pas la fenêtre. Vous devez programmer un gestionnaire appelant `Window.deletePopUp()` pour fermer la fenêtre explicitement. Pour plus d'informations sur l'événement `click`, consultez `Window.click`.

Vous pouvez rédiger du code ActionScript pour contrôler ces options et d'autres options des composants Window en utilisant les propriétés, méthodes et événements Actionscript. Pour plus d'informations, consultez Classe Window.

Création d'une application avec le composant Window

La procédure suivante explique comment ajouter un composant Window à une application. Dans cet exemple, l'utilisateur est invité, via la fenêtre, à modifier son mot de passe et à confirmer le nouveau.

Pour créer une application avec le composant Button, procédez comme suit :

1. Créez un nouveau clip contenant des champs de mot de passe et de confirmation de mot de passe, ainsi que des boutons OK et Annuler. Nommez le clip **FormulaireMotDePasse**.
2. Dans la bibliothèque, sélectionnez le clip FormulaireMotDePasse et choisissez Liaison dans le menu d'options.
3. Activez l'option Exporter pour ActionScript et entrez **FormulaireMotDePasse** dans le champ Identifiant.
4. Entrez **mx.core.View** dans le champ de classe.
5. Faites glisser un composant Window du panneau Composants jusqu'à la scène et supprimez le composant de la scène. Cette action permet d'ajouter le composant à la bibliothèque.
6. Dans la bibliothèque, sélectionnez le composant Window SWC et choisissez Liaison dans le menu d'options.
7. Activez l'option Exporter pour ActionScript.
8. Ouvrez le panneau Actions et entrez le gestionnaire de clic suivant sur l'image 1 :

```
buttonListener = new Object();
buttonListener.click = function(){
  mx.managers.PopUpManager.createPopUp(_root, mx.containers.Window, true, {
  title:"Modifier mot de passe", contentPath:"FormulaireMotDePasse" })
}
button.addEventListener("click", buttonListener);
```

Ce gestionnaire appelle `PopUpManager.createPopUp()` pour instancier un composant Window, avec la barre de titre « Modifier mot de passe », qui affiche le contenu du clip FormulaireMotDePasse.

Personnalisation du composant Window

Vous pouvez transformer un composant Window horizontalement et verticalement au cours de la programmation comme à l'exécution. Lors de la création, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez `UIObject.setSize()` ou toute propriété et méthode applicable de la classe Window. Pour plus d'informations, consultez Classe Window.

La modification de la taille de la fenêtre ne change pas la taille du bouton ou du titre. Le titre est aligné sur la gauche et la barre de fermeture sur la droite.

Utilisation de styles avec le composant Window

La déclaration de style de la barre de titre d'un composant Window est indiquée par la propriété `Window.titleStyleDeclaration`.

Un composant Window supporte les styles de halo suivants :

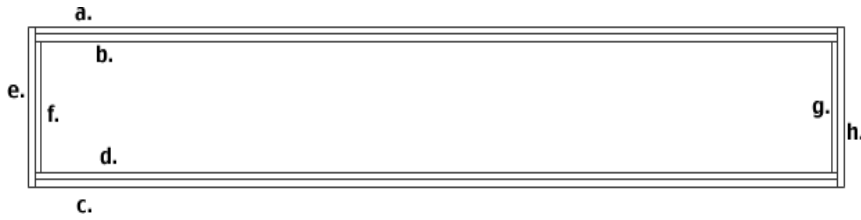
Style	Description
borderStyle	Bordure du composant : "none", "inset", "outset" ou "solid". Ce style n'hérite pas de sa valeur.

Utilisation d'enveloppes avec le composant Window

Le composant Window utilise la classe RectBorder qui fait appel à l'API de dessin ActionScript pour dessiner ses bordures. La méthode `setStyle()` (consultez `UIObject.setStyle()`) vous permet de modifier les propriétés suivantes du style RectBorder :

Styles RectBorder
`borderColor`
`highlightColor`
`borderColor`
`shadowColor`
`borderCapColor`
`shadowCapColor`
`shadowCapColor`
`borderCapColor`

Les propriétés de style définissent les positions suivantes sur la bordure :



Si vous utilisez `UIObject.createClassObject()` ou `PopUpManager.createPopUp()` pour créer une occurrence Window de façon dynamique (à l'exécution), vous pouvez également lui appliquer une enveloppe de façon dynamique. Pour appliquer un composant lors de l'exécution, définissez les propriétés d'enveloppe du paramètre *`initObject`* qui est passé à la méthode `createClassObject()`. Ces propriétés d'enveloppe définissent les noms des symboles à utiliser en tant qu'états du bouton, avec et sans icône. Pour plus d'informations, consultez `UIObject.createClassObject()` et `PopUpManager.createPopUp()`.

Un composant Window utilise les propriétés d'enveloppe suivantes :

Propriété	Description
`skinTitleBackgroud`	La barre de titre. La valeur par défaut est `TitleBackground`.
`skinCloseUp`	Le bouton Fermer. La valeur par défaut est `CloseButtonUp`.

Propriété	Description
skinCloseDown	Le bouton Fermer à l'état enfoncé. La valeur par défaut est CloseButtonDown.
skinCloseDisabled	Le bouton Fermer à l'état désactivé. La valeur par défaut est CloseButtonDisabled.
skinCloseOver	Le bouton Fermer à l'état au-dessus. La valeur par défaut est CloseButtonOver.

Classe Window

Héritage UIObject > UIComponent > View > ScrollView > Window

Espace de nom de classe ActionScript mx.containers.Window

Les propriétés de la classe Window vous permettent de définir le titre, d'ajouter un bouton Fermer et de définir le contenu d'affichage à l'exécution. La définition d'une propriété de la classe Window avec ActionScript annule le paramètre du même nom défini dans le panneau de l'inspecteur des propriétés ou des composants.

La meilleure façon d'instancier une fenêtre est d'appeler PopUpManager.createPopUp(). Cette méthode crée une fenêtre qui peut être modale (chevauchant et désactivant des objets existants dans une application) ou non modale. Le code suivant, par exemple, crée une occurrence de fenêtre modale (ce que dénote le dernier paramètre) :

```
var nouvelleFenêtre = PopUpManager.createPopUp(this, Window, true);
```

La création d'une grande fenêtre transparente sous le composant Window permet de simuler la modalité. En raison de la façon selon laquelle les fenêtres transparentes apparaissent, les objets qui se trouvent en dessous peuvent apparaître légèrement estompés. L'effet de transparence en cours peut être modifié en faisant varier la valeur _global.style.modalTransparency entre 0 (complètement transparent) et 100 (opaque). Si vous faites en sorte que la fenêtre soit partiellement transparente, vous pouvez également définir la couleur de la fenêtre en modifiant l'enveloppe Modal dans le thème par défaut.

Si vous utilisez PopUpManager.createPopUp() pour créer une fenêtre modale, vous devez appeler Window.deletePopUp() pour la supprimer afin que la fenêtre transparente soit également supprimée. Par exemple, si vous utilisez closeButton sur la fenêtre, vous écrirez le code suivant :

```
obj.click = function(evt){
   this.deletePopUp();
}
window.addEventListener("click", obj);
```

Remarque : Le code n'interrompt pas l'exécution lorsqu'une fenêtre modale est créée. Dans d'autres environnements, Microsoft Windows par exemple, si vous créez une fenêtre modale, les lignes de code qui suivent la création de la fenêtre ne sont pas exécutées tant que la fenêtre n'est pas fermée. Dans Flash, les lignes de code sont exécutées après la création de la fenêtre et avant sa fermeture.

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

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

Remarque : Le code suivant renvoie la valeur undefined :
`trace(monOccurrenceFenêtre.version);`.

Méthodes de la classe Window

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

Hérite de toutes les méthodes de la Classe UIObject, Classe UIComponent et View.

Propriétés de la classe Window

Propriété	Description
`Window.closeButton`	Indique si un bouton Fermer est inclus dans la barre de titre (`true`) ou non (`false`).
`Window.content`	Une référence au contenu spécifié dans la propriété `contentPath`.
`Window.contentPath`	Un chemin qui mène au contenu affiché dans la fenêtre.
`Window.title`	Le texte qui s'affiche dans la barre de titre.
`Window.titleStyleDeclaration`	La déclaration de style qui formate le texte dans la barre de titre.

Hérite de toutes les propriétés de Classe UIObject, Classe UIComponent et ScrollView.

Evénements de la classe Window

Evénement	Description
`Window.click`	Déclenché lorsque le bouton Fermer est relâché.
`Window.mouseDownOutside`	Déclenché lorsque l'utilisateur clique sur le bouton de la souris en dehors de la fenêtre modale.

Hérite de tous les événements de la Classe UIObject, de la Classe UIComponent, View et ScrollView.

Window.click

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

```
ObjetDécoute = new Object();
ObjetDécoute.click = function(eventObject){
  ...
}
occurrenceFenêtre.addEventListener("click", objetDécoute)
```

Description

Evénement : diffusion à tous les écouteurs enregistrés lorsque l'utilisateur clique sur le bouton Fermer (puis relâche le bouton de la souris).

Le premier exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence de composant Window. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de comportement. Par exemple, le code suivant, associé à l'occurrence de composant Window `maFenêtre`, envoie « _level0.maFenêtre » au panneau de sortie :

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`occurrenceFenêtre`*) distribue un événement (ici, `click`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. L'objet événement a un jeu de propriétés contenant des informations sur l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

L'exemple suivant crée une fenêtre modale, puis définit un gestionnaire de clic pour la supprimer : Vous devez ajouter un composant Window à la scène, puis le supprimer pour l'ajouter à la bibliothèque de documents et, enfin, insérer le code suivant dans l'image 1 :

```
import mx.managers.PopUpManager
import mx.containers.Window
var maTW = PopUpManager.createPopUp(_root, Window, true, {closeButton: true,
  title:"Ma fenêtre"});
windowListener = new Object();
windowListener.click = function(evt){
  _root.maTW.deletePopUp();
}
maTW.addEventListener("click", windowListener);
```

Consultez également

`UIEventDispatcher.addEventListener()`, `Window.closeButton`

Window.closeButton

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceFenêtre.closeButton

Description

Propriété : valeur booléenne qui indique si la barre de titre doit avoir un bouton Fermer (`true`) ou non (`false`). Cette propriété doit être définie dans le paramètre *objetInit* de la méthode `PopUpManager.createPopUp()`. La valeur par défaut est `false`.

Exemple

Le code suivant crée une fenêtre qui affiche le contenu dans le clip « FormulaireDeConnexion » et qui a un bouton Fermer dans la barre de titre :

```
var maTW = PopUpManager.createPopUp(_root, Window, true,
  {contentPath:"FormulaireDeConnexion", closeButton:true});
```

Consultez également

`Window.click`, `PopUpManager.createPopUp()`

Window.content

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceFenêtre.content

Description

Propriété : référence au contenu (clip racine) de la fenêtre. Cette propriété renvoie un objet MovieClip. Lorsque vous associez un symbole de la bibliothèque, la valeur par défaut est une occurrence de ce symbole. Lorsque vous chargez du contenu à partir d'une URL, la valeur par défaut est undefined jusqu'à ce que le chargement commence.

Exemple

Définissez la valeur de la propriété text dans le contenu à l'intérieur du composant Window :

```
loginForm.content.password.text = "secret";
```

Window.contentPath

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceFenêtre.contentPath

Description

Propriété : définit le nom du contenu à afficher dans la fenêtre. Cette valeur peut être l'identificateur de liaison d'un clip de la bibliothèque ou une URL absolue ou relative du fichier SWF ou JPG à charger. La valeur par défaut est "" (chaîne vide).

Exemple

Le code suivant crée une occurrence Window qui affiche le clip dont l'identificateur de liaison correspond à « FormulaireDeConnexion » :

```
var maTW = PopUpManager.createPopUp(_root, Window, true,
  {contentPath:"FormulaireDeConnexion"});
```

Window.deletePopUp()

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

occurrenceFenêtre.deletePopUp();

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : supprime l'occurrence Window et supprime l'état modal. Cette méthode peut uniquement être appelée sur des occurrences Window qui ont été créées par PopUpManager.createPopUp().

Exemple

Le code suivant crée une fenêtre modale, puis crée un écouteur qui supprime la fenêtre lorsque l'utilisateur clique sur le bouton Fermer :

```
var maTW = PopUpManager.createPopUp(_root, Window, true);
twListener = new Object();
```

```
twListener.click = function(){
  maTW.deletePopUp();
}
maTW.addEventListener("click", twListener);
```

Window.mouseDownOutside

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(mouseDownOutside){
  ...
}
```

Usage 2 :

```
ObjetDécoute = new Object();
ObjetDécoute.mouseDownOutside = function(objetEvnt){
  ...
}
occurrenceFenêtre.addEventListener("mouseDownOutside", objetDécoute)
```

Description

Evénement : diffusion à tous les écouteurs enregistrés lorsque l'utilisateur clique sur le bouton de la souris (puis le relâche) en dehors de la fenêtre modale. Cet événement est rarement utilisé, mais vous pouvez vous en servir pour fermer une fenêtre si l'utilisateur essaie d'interagir avec un élément situé à l'extérieur de cette fenêtre.

Le premier exemple d'utilisation fait appel au gestionnaire `on()` et doit être associé directement à une occurrence de composant Window. Le mot-clé `this`, utilisé dans un gestionnaire `on()` lié à un composant, correspond à l'occurrence de comportement. Par exemple, le code suivant, associé à l'occurrence de composant Window `monComposantWindow`, envoie « _level0.monComposantWindow » au panneau de sortie :

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`occurrenceFenêtre`*) distribue un événement (ici, `mouseDownOutside`) qui est géré par une fonction associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous définissez une méthode portant le même nom que l'événement traité par l'objet d'écoute ; la méthode est appelée lorsque l'événement est déclenché. Lorsque l'événement est déclenché, il passe automatiquement un objet événement (*`objetEvnt`*) à la méthode d'objet d'écoute. L'objet événement a un jeu de propriétés contenant des informations sur l'événement. Vous pouvez utiliser ces propriétés pour rédiger le code qui traitera l'événement. Pour finir, vous appelez la méthode `UIEventDispatcher.addEventListener()` sur l'occurrence de composant qui diffuse l'événement pour enregistrer l'écouteur dans l'occurrence. Lorsque l'occurrence distribue l'événement, l'écouteur approprié est appelé.

Pour plus d'informations sur les objets événements, consultez *Objets événement*, page 240.

Exemple

L'exemple suivant crée une occurrence Window et définit un gestionnaire `mouseDownOutside` qui appelle une méthode `beep()` si l'utilisateur clique en dehors de la fenêtre :

```
var maTW = PopUpManager.createPopUp(_root, Window, true, undefined, true);
// création d'un écouteur
twListener = new Object();
twListener.mouseDownOutside = function()
{
    beep(); // émet un son si l'utilisateur clique en dehors de la fenêtre
}
maTW.addEventListener("mouseDownOutside", twListener);
```

Consultez également

`UIEventDispatcher.addEventListener()`

Window.title

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`occurrenceFenêtre``.title`

Description

Propriété : chaîne correspondant au titre de la barre de titre. La valeur par défaut est `""` (chaîne vide).

Exemple

Le code suivant définit le titre de la fenêtre à « Bonjour Monde » :

```
maTW.title = "Bonjour Monde";
```

Window.titleStyleDeclaration

Disponibilité

Flash Player 6.0.79.

Edition

Flash MX 2004.

Usage

`occurrenceFenêtre``.titleStyleDeclaration`

Description

Propriété : chaîne indiquant la déclaration de style qui formate la barre de titre d'une fenêtre. La valeur par défaut n'est pas définie, ce qui correspond à gras et texte blanc.

Exemple

Le code suivant crée une fenêtre qui affiche le contenu du clip doté de l'identificateur de liaison « ModifierMotDePasse » et qui utilise la déclaration de style CSS « MesStylesTW » :

```
var maTW = PopUpManager.createPopUp(_root, Window, true,
  {contentPath:"FormulaireDeConnexion",
                                    titleStyleDeclaration:"MesStylesTW"});
```

Pour plus d'informations sur les styles, consultez *Utilisation des styles pour personnaliser la couleur et le texte des composants*, page 27.

Composant XMLConnector

Pour des informations actualisées sur cette fonction, cliquez sur le bouton Mettre à jour dans la partie supérieure de l'onglet Aide.

Composant XUpdateResolver

Pour des informations actualisées sur cette fonction, cliquez sur le bouton Mettre à jour dans la partie supérieure de l'onglet Aide.

CHAPITRE 5
Création de composants

Ce chapitre indique comment créer des composants utilisables par d'autres développeurs et comment les préparer pour le déploiement.

Nouveautés

La version actuelle (version 2) de l'architecture des composants Macromedia diffère beaucoup de la version de Macromedia Flash MX (version 1). Les améliorations apportées ont permis d'optimiser l'évolutivité, la performance et l'extensibilité des composants pour les développeurs. La liste suivante présente une partie de ces améliorations :

- le panneau Inspecteur de composants reconnaît les métadonnées ActionScript ;
- gestionnaires et classes de données extensibles ;
- aperçu en direct intégré ;
- messages de compilation améliorés ;
- nouveau modèle d'événement ;
- gestion focus ;
- styles CSS.

Travail dans l'environnement Flash

L'environnement de Macromedia Flash MX 2004 et Flash MX Professionnel 2004 est conçu pour fournir une structure logique de classes et de composants. Cette section explique où stocker les fichiers de composants.

Fichiers FLA

Lorsque vous créez un composant, vous commencez par un fichier FLA, puis vous ajoutez des enveloppes, des graphiques et d'autres éléments. Vous pouvez stocker ces éléments partout dans le fichier FLA, parce que l'utilisateur de composants Flash travaille avec un fichier de composants compilé, pas avec les actifs source.

Lors de la création de composants dans Flash, vous utilisez des fichiers SWF à image et à calque doubles. Le premier calque est un calque d'actions qui désigne le fichier de classe ActionScript du composant. Le second calque est un calque d'actifs contenant les graphiques, symboles et autres éléments utilisés par le composant.

Fichiers de classe

Le fichier FLA inclut une référence au fichier de classe ActionScript pour le composant. On parle alors de liaison du composant au fichier de classe.

Le code ActionScript spécifie les propriétés et méthodes du composant et définit les classes dont le composant hérite, le cas échéant. Utilisez la convention d'affectation de nom *.as pour le code source ActionScript et attribuez au fichier de code source le nom du composant. Par exemple, MonComposant.as contient le code source du composant MonComposant.

Les fichiers Flash MX 2004 .as de classe de base résident dans un même dossier appelé Classes/mx/Core. Les autres fichiers de classe ActionScript sont organisés par nom de paquet dans leurs dossiers respectifs, sous /Classes.

Dans le cas d'un composant personnalisé, créez un répertoire sous /Classes et stockez-y le fichier de classe ActionScript.

Le chemin de classe

Cette section décrit le chemin de classe de Flash.

Présentation du chemin de classe

Le chemin de classe est une liste triée de répertoires dans laquelle Flash recherche les fichiers de classe lors de l'exportation d'un composant ou de la génération d'un fichier SWF. L'ordre des entrées du chemin de classe est important parce que Flash utilise la première classe trouvée lors d'une recherche. Lors d'une exportation, les classes trouvées dans le chemin de classe et qui correspondent aux identificateurs de liaison dans le fichier FLA sont importées dans ce dernier et enregistrées avec leurs symboles.

Un chemin de classe global fait référence à tous les fichiers FLA générés par Flash. Un chemin de classe local s'applique uniquement au fichier FLA actif.

Le chemin de classe local par défaut est vide. Le chemin de classe global est constitué des deux chemins suivants :

- $(UserConfig)/Classes
- .

Le point (.) indique le répertoire actif. Flash recherche les classes ActionScript dans le répertoire courant du fichier FLA.

Le chemin $(UserConfig)/Classes indique le répertoire de configuration par utilisateur. Ce répertoire désigne les emplacements suivants :

- Sous Windows, il s'agit du répertoire c:\Documents and Settings*nomDutilisateur*\Application Data\Macromedia\Flash MX 2004\en\Configuration.
- Sous Macintosh, il s'agit du répertoire *volume*:Users:*nomDutilisateur*:Library:Application Support:Macromedia:Flash MX 2004:en:configuration.

Les répertoires UserConfig reflètent les répertoires situés sous *Flash_root*/en/Configuration. Toutefois, le chemin de classe n'inclut pas directement ces répertoires et il est relatif au répertoire UserConfig.

Modification du chemin de classe

Vous pouvez modifier le chemin de classe d'un fichier FLA particulier (chemin de classe local) ou de tous les fichiers FLA que vous utilisez dans Flash (chemin de classe global).

Pour modifier le chemin de classe global :

1 Sélectionnez Edition > Préférences.

La boîte de dialogue Préférences s'affiche.

2 Sélectionnez l'onglet ActionScript.

3 Cliquez sur le bouton Paramètres d'ActionScript 2.0.

La boîte de dialogue Paramètres d'ActionScript s'affiche.

4 Ajoutez, supprimez ou modifiez des entrées dans la zone Chemin de classe.

5 Enregistrez vos modifications.

Pour modifier le chemin de classe local :

1 Choisissez Fichier > Paramètres de publication.

La boîte de dialogue du même nom s'affiche.

2 Sélectionnez l'onglet Flash.

3 Cliquez sur le bouton Paramètres.

La boîte de dialogue Paramètres d'ActionScript s'affiche.

4 Ajoutez, supprimez ou modifiez des entrées dans la zone Chemin de classe.

5 Enregistrez vos modifications.

Recherche de fichiers source de composants

Lorsque vous créez un composant, vous pouvez stocker les fichiers source dans le répertoire de votre choix. Vous devez cependant inclure ce répertoire dans les paramètres de chemin de classe Flash MX 2004 de sorte que Flash puisse trouver les fichiers de classe requis lors de l'exportation du composant. En outre, pour pouvoir être testé, le composant doit être stocké dans le répertoire Flash Components. Pour plus d'informations sur le stockage des fichiers SWC, consultez *Utilisation de fichiers SWC*, page 294.

Manipulation des symboles

Chaque symbole possède son propre scénario. Vous pouvez y ajouter des images, des images-clés et des calques comme dans le scénario principal.

Lorsque vous créez des composants, vous commencez par un symbole. Flash permet de manipuler un symbole à l'aide des trois méthodes suivantes :

- Manipulez le symbole dans le contexte des autres objets de la scène à l'aide de la commande Modifier en place. Les autres objets apparaissent en grisé pour les distinguer du symbole que vous modifiez. Le nom du symbole que vous manipulez est affiché dans une barre d'édition située en haut de la scène, à droite du nom de la séquence courante.
- Manipulez le symbole dans une fenêtre séparée à l'aide de la commande Modifier dans une nouvelle fenêtre. La modification d'un symbole dans une autre fenêtre vous permet de visualiser le symbole et le scénario principal de façon simultanée. Le nom du symbole que vous manipulez est affiché dans une barre d'édition, en haut de la scène.

- Manipulez le symbole en changeant la fenêtre de façon à ne plus afficher la scène mais seulement le symbole (dans le mode d'édition de symbole). Le nom du symbole que vous manipulez est affiché dans une barre d'édition, en haut de la scène, à droite du nom de la séquence courante.

Exemples de code de composant

Flash MX 2004 et Flash MX Professionnel 2004 incluent les fichiers source de composant suivants afin de faciliter la création de composants :

- Code source de fichier FLA : *Flash MX 2004_install_dir*/en/First Run/ComponentFLA/StandardComponents.fla
- Fichiers de classe ActionScript : *Flash MX 2004_install_dir*/en/First Run/Classes/mx

Création de composants

Cette section décrit la création d'un composant dans une sous-classe d'une classe Flash MX 2004 existante. Les sections suivantes expliquent comment rédiger le fichier de classe ActionScript du composant et comment modifier le composant pour obtenir une fonctionnalité et une qualité optimales.

Création d'un symbole de composant

Tous les composants sont des objets MovieClip, qui sont un type de symbole. Pour créer un composant, vous devez d'abord insérer un nouveau symbole dans un nouveau fichier FLA.

Pour ajouter un symbole de composant :

1. Dans Flash, créez un document Flash vierge.
2. Sélectionnez Insertion > Nouveau symbole.

   La boîte de dialogue Créer un nouveau symbole s'affiche.
3. Saisissez un nom de symbole. Attribuez un nom au composant en commençant chaque mot par une majuscule (par exemple, MonComposant).
4. Sélectionnez le bouton radio Clip pour le comportement.

   Un objet Clip possède son propre scénario, qui est lu indépendamment du scénario principal.
5. Cliquez sur le bouton Avancé.

   Les paramètres avancés s'affichent dans la boîte de dialogue.
6. Choisissez Exporter pour ActionScript. Cette option indique au logiciel d'inclure le composant par défaut au contenu Flash qui utilise le composant.
7. Saisissez un identificateur de liaison.

   Cet identificateur est utilisé pour le nom de symbole, le nom de liaison et le nom de classe associée.
8. Dans la zone de texte Classe AS 2.0, entrez le chemin entièrement qualifié de la classe ActionScript 2.0, relatif aux paramètres de chemin de classe.

   Remarque : N'incluez pas l'extension du fichier. La zone de texte Classe AS 2.0 désigne l'emplacement du paquet de la classe, pas le nom du fichier dans le système de fichiers.

Si le fichier ActionScript est dans un paquet, vous devez inclure le nom de ce dernier. La valeur de ce champ peut être relative au chemin de classe ou être un chemin de paquet absolu (par exemple, monPaquet.MonComposant).

Pour plus d'informations sur la configuration du chemin de classe Flash MX 2004, consultez *Présentation du chemin de classe*, page 270.

9 En règle générale, vous devez désélectionner l'option Exporter dans la première image, qui est sélectionnée par défaut. Pour plus d'informations, consultez *Recommandations en matière de conception d'un composant*, page 296.

10 Cliquez sur OK.

Flash ajoute le symbole à la bibliothèque et passe en mode d'édition de symbole. Dans ce mode, le nom du symbole apparaît au-dessus de l'angle supérieur gauche de la scène et une mire indique le point d'alignement du symbole.

Vous pouvez désormais manipuler le symbole et l'ajouter au fichier FLA de votre composant.

Manipulation de calques de symboles

Une fois le symbole créé et ses liaisons définies, vous pouvez déterminer les actifs du composant dans le scénario du symbole.

Un symbole de composant doit avoir deux calques. Cette section décrit les calques à insérer et ce qu'il faut y ajouter.

Pour plus d'informations sur la manipulation de symboles, consultez *Manipulation des symboles*, page 271.

Pour manipuler des calques de symboles :

1 Entrez en mode d'édition de symbole.

2 Renommez un calque vide ou créez un calque appelé Actions.

3 Dans le panneau Actions, ajoutez une ligne unique pour importer le fichier de classe ActionScript entièrement qualifié du composant.

Cette instruction s'appuie sur les paramètres de chemin de classe Flash MX 2004. Pour plus d'informations, consultez *Présentation du chemin de classe*, page 270. L'exemple suivant importe le fichier MonComposant.as situé dans le paquet monPaquet :

```
import monPaquet.MonComposant;
```

Remarque : Utilisez l'instruction `import`, pas `include`, lors de l'importation d'un fichier de classe ActionScript. N'entourez pas le nom de classe ou le paquet de points d'interrogation.

4 Renommez un calque vide ou créez un calque appelé Actifs.

Le calque Actifs inclut tous les actifs utilisés par ce composant.

5 Dans la première image, ajoutez une action `stop()` dans le panneau Actions, comme dans l'exemple suivant :

```
stop();
```

N'ajoutez pas d'actif graphique à cette image. Flash Player s'arrêtera avant la deuxième image, dans laquelle vous pouvez ajouter des actifs.

6 Si vous étendez un composant existant, recherchez ce dernier ainsi que toutes les classes de base que vous utilisez, puis placez une occurrence de ce symbole dans la seconde image de votre calque. Pour cela, sélectionnez le symbole dans le panneau Composants et faites-le glisser sur la scène de la seconde image du calque Actifs du composant.

Tous les actifs utilisés par un composant (qu'il s'agisse d'un autre composant ou d'éléments tels que des bitmaps) doivent posséder une occurrence, placée dans le composant.

7 Ajoutez les actifs graphiques utilisés par ce composant à la seconde image du calque Actifs de votre composant. Par exemple, si vous créez un bouton personnalisé, ajoutez les graphiques représentant les états du bouton (haut, bas, etc.).

8 Une fois le contenu du symbole créé, effectuez l'une des opérations suivantes pour revenir au mode d'édition de document :

- Cliquez sur le bouton de retour, du côté gauche de la barre d'édition au-dessus de la scène.
- Sélectionnez Edition > Modifier le document.
- Cliquez sur le nom de la séquence dans la barre d'édition au-dessus de la scène.

Ajout de paramètres

La prochaine étape du développement du composant consiste à définir ses paramètres. Les paramètres constituent la première méthode de modification des occurrences des composants que vous avez créés.

Dans les versions précédentes de Flash, les paramètres étaient définis dans le panneau Inspecteur de composants. Dans Flash MX 2004 et Flash MX Professionnel 2004, les paramètres sont définis dans le fichier de classe ActionScript. Le panneau Inspecteur de composants détecte les paramètres publics et les affiche dans l'interface utilisateur.

La section suivante traite de la rédaction du fichier ActionScript externe, qui inclut des informations sur l'ajout de paramètres de composant.

Rédaction d'ActionScript du composant

La plupart des composants comprennent un code ActionScript. Le type de composant détermine l'emplacement et la quantité de la rédaction du code ActionScript. Un composant peut être développé de deux manières principales :

- création de nouveaux composants sans classe parent ;
- extension de classes de composant existantes.

Cette section explique comment étendre des composants existants. Si vous créez un composant dérivé d'un fichier de classe d'un autre composant, vous devez rédiger un fichier de classe ActionScript externe, comme indiqué dans cette section.

Extension de classes de composant existantes

Lorsque vous créez un symbole de composant dérivé d'une classe parent, vous le liez à un fichier de classe ActionScript 2.0 externe. Pour plus d'informations sur la définition de ce fichier, consultez *Création d'un symbole de composant*, page 272.

La classe ActionScript externe étend une autre classe, ajoute des méthodes, des lectures et des définitions et définit des gestionnaires d'événement pour le composant. Pour modifier des fichiers de classe ActionScript, vous pouvez utiliser Flash, un éditeur de texte ou un environnement de développement intégré (IDE - Integrated Development Environment).

Vous pouvez hériter d'une seule classe. ActionScript 2.0 n'autorise pas les héritages multiples.

Exemple simple d'un fichier de classe

L'exemple suivant présente un fichier de classe appelé MonComposant.as. Il contient un jeu minimal d'importations, de méthodes et de déclarations pour un composant héritant de la classe UIObject.

```
import mx.core.UIObject;

  class monPaquet.MonComposant extends UIObject {
  static var nomSymbole:String = "MonComposant";
  static var proprietaireSymbole:Object = Object(monPaquet.MonComposant);
  var className:String = "MonComposant";
  #include "../core/VersionDuComposant.as"
  function MonComposant() {
  }
  function init(Void):Void {
    super.init();
  }
  function size(Void):Void {
    super.size();
  }
}
```

Processus général de rédaction d'un fichier de classe

Utilisez le processus général suivant lors de la rédaction du fichier ActionScript pour un composant. Selon le type de composant que vous créez, certaines étapes peuvent être superflues.

Ce processus est traité plus en détail à la fin de ce chapitre.

Pour rédiger le fichier ActionScript d'un composant :

1 Importez toutes les classes requises.

2 Définissez la classe à l'aide du mot-clé `class` et étendez une classe parent au besoin.

3 Définissez les variables `nomSymbole` et `proprietaireSymbole` ; ce sont les noms respectifs des symboles de votre classe ActionScript et du paquet entièrement qualifié de la classe.

4 Définissez votre nom de classe comme la variable `nomClasse`.

5 Ajoutez des informations de versionnage.

6 Entrez vos variables de membre par défaut.

7 Créez des variables pour chacun des éléments d'enveloppe et des liaisons du composant. Cela permet aux utilisateurs de définir un élément d'enveloppe différent en modifiant un paramètre du composant.

8 Ajoutez des constantes de classe.

9 Ajoutez un mot-clé et une déclaration de métadonnées pour chaque variable possédant une lecture et une définition.

10 Définissez les variables de membre non initialisé.

11 Définissez les lectures et les définitions.

12 Rédigez un constructeur. En règle générale, ce dernier doit être vide.

13 Ajoutez une méthode d'initialisation. Cette méthode est appelée lorsque la classe est créée.

14 Ajoutez une méthode de taille.

15 Ajoutez des méthodes personnalisées ou remplacez les méthodes héritées.

Importation de classes

La première ligne de votre fichier de classe ActionScript externe doit importer les fichiers de classe nécessaires utilisés par votre classe. Cette opération inclut les classes qui fournissent une fonctionnalité ainsi que la superclasse étendue par votre classe, le cas échéant.

Lorsque vous utilisez l'instruction `import`, vous importez le nom de classe entièrement qualifié, plutôt que le nom de fichier de la classe, comme l'illustre l'exemple suivant :

```
import mx.core.UIObject;
import mx.core.ScrollView;
import mx.core.ext.UIObjectExtensions;
```

Vous pouvez également utiliser le caractère générique (*) pour importer toutes les classes dans un paquet donné. Par exemple, l'instruction suivante importe toutes les classes dans le paquet `mx.core` :

```
import mx.core.*;
```

Si une classe importée n'est pas utilisée dans un script, la classe n'est pas incluse dans le pseudo-code binaire résultant du fichier SWF. Ainsi, l'importation d'un paquet entier avec un caractère générique ne crée pas de fichier SWF trop volumineux.

Sélection d'une classe parent

La plupart des composants possèdent un comportement et une fonctionnalité communs. Flash fournit à cet effet deux classes de base. En sous-classant ces classes, votre composant commence par un jeu de base de méthodes, de propriétés et d'événements.

Le tableau suivant présente ces deux classes de base :

Classe entière	Etend	Description
mx.core.UIObject	MovieClip	UIObject est la classe de base de tous les objets graphiques. Elle peut être dotée d'une forme, se dessiner et être invisible. UIObject fournit les fonctionnalités suivantes : • Modification de styles • Gestion d'événements • Redimensionnement par mise à l'échelle
mx.core.UIComponent	UIObject	UIComponent est la classe de base de tous les composants. Elle peut participer à la tabulation, accepter les événements de bas niveau tels que la saisie au clavier et les actions de la souris, et être désactivée afin d'empêcher ces deux dernières opérations. UIComponent fournit les fonctionnalités suivantes : • Création de la navigation du focus • Activation et désactivation des composants • Redimensionnement des composants

Présentation de la classe UIObject

Les composants basés sur la version 2 de l'architecture des composants Macromedia sont basés sur la classe UIObject, qui englobe la classe MovieClip. La classe MovieClip est la classe de base de toutes les classes de Flash permettant de dessiner à l'écran. De nombreuses propriétés et méthodes MovieClip sont associées au Scénario, un outil que les développeurs découvrant Flash ne connaissent pas. La classe UIObject a été conçue pour omettre la plupart de ces détails. Les sous-classes de MovieClip n'affichent pas les propriétés et méthodes superflues. Vous pouvez cependant y accéder si vous le souhaitez.

UIObject essaie de masquer la gestion de la souris et de l'image dans MovieClip. Il publie des événements à ces écouteurs juste avant le dessin (l'équivalent de `onEnterFrame`), lors du chargement et du déchargement, et lorsque la disposition est modifiée (`move`, `resize`).

UIObject fournit des variables en lecture seule différentes permettant de déterminer la position et la taille du clip. Vous pouvez utiliser les méthodes `move()` et `setSize()` pour modifier la position et la taille d'un objet.

Présentation de la classe UIComponent

La classe UIComponent est un enfant de UIObject. Elle constitue la base de tous les composants possédant une interaction avec l'utilisateur (action de la souris et saisie clavier).

Extension d'autres classes

Afin de faciliter la construction d'un composant, vous pouvez sous-classer une classe au lieu d'étendre directement la classe UIObject ou UIComponent. Si vous étendez la classe d'un autre composant, vous étendez également ces classes par défaut. Toutes les classes répertoriées dans le dictionnaire des composants peuvent être étendues pour créer une nouvelle classe de composant.

Flash comprend un groupe de classes qui dessinent à l'écran et héritent de la classe UIObject. Par exemple, la classe Border dessine les bordures autour des autres objets. De même, RectBorder, une sous-classe de Border, sait comment redimensionner ses éléments visuels correctement. Tous les composants gérant les bordures doivent utiliser l'une des classes ou sous-classes de bordures. Pour une description détaillée de ces classes, consultez le Chapitre 4, *Dictionnaire des composants*, page 47.

Par exemple, si vous voulez créer un composant qui se comporte presque exactement comme un composant Button, vous pouvez étendre la classe Button au lieu d'en recréer toutes les fonctionnalités à partir des classes de base.

Rédaction du constructeur

Les constructeurs sont des méthodes possédant un objectif unique : définir les propriétés et effectuer d'autres tâches lorsqu'une nouvelle instruction de composant est instanciée.
Un constructeur se reconnaît à son nom : il est identique au nom de la classe du composant.
Par exemple, le code suivant indique le constructeur du composant ScrollBar :

```
fonction ScrollBar() {
}
```

Dans ce cas, lorsqu'un nouveau composant ScrollBar est instancié, le constructeur `ScrollBar()` est appelé.

Il est généralement préférable de laisser les constructeurs de composant vides pour que l'objet puisse être personnalisé avec son interface de propriétés. En outre, la définition des propriétés d'un constructeur peut entraîner l'écrasement des valeurs par défaut, selon l'ordre des appels d'initialisation.

Une classe ne peut contenir qu'une fonction constructeur ; les fonctions constructeur surchargées ne sont pas autorisées dans ActionScript 2.0.

Versionnage

Lorsque vous publiez un nouveau composant, vous devez définir un numéro de version. Ainsi, les développeurs savent qu'ils doivent effectuer une mise à niveau et le support technique en est facilité. Pour définir un numéro de version, utilisez la variable statique `version`, comme dans l'exemple suivant :

```
static var version:String = "1.0.0.42";
```

Si vous créez plusieurs composants dans un paquet de composants, vous pouvez inclure le numéro de version dans un fichier externe. Cela permet de mettre à jour le numéro de version une seule fois. Par exemple, le code suivant importe le contenu d'un fichier externe qui stocke le numéro de version en un seul emplacement :

```
#include "../monPaquet/VersionDuComposant.as"
```

Le contenu du fichier VersionDuComposant.as est identique à la déclaration de variable ci-dessus, comme dans l'exemple suivant :

```
static var version:String = "1.0.0.42";
```

Noms de la classe, du symbole et du propriétaire

Pour aider Flash à localiser les classes ActionScript et les paquets appropriés et à préserver l'affectation des noms du composant, vous devez définir les propriétés `nomSymbole`, `proprietaireSymbole` et `nomClasse` dans le fichier de classe ActionScript de votre composant.

Le tableau suivant présente ces variables :

Variable	Description
`nomSymbole`	Nom du symbole de l'objet. Cette variable est statique et de type String.
`proprietaireSymbole`	Classe utilisée dans l'appel interne à la méthode `createClassObject()`. Cette valeur doit être le nom de classe entièrement qualifié, qui inclut le chemin du paquet. Cette variable est statique et de type Object.
`nomClasse`	Nom de la classe du composant. Cette variable est également utilisée lors du calcul des valeurs de style. Si `_global.styles[nomClasse]` existe, elle définit les valeurs par défaut d'un composant. Cette variable est de type String.

L'exemple suivant illustre l'affectation d'un nom à un composant personnalisé :

```
static var nomSymbole:String = "MonComposant";
static var proprietaireSymbole:Object = custom.MonComposant;
var nomClasse:String = "MonComposant";
```

Définition de lectures et de définitions

Les méthodes get et set permettent de connaître les propriétés des composants et de contrôler l'accès à ces propriétés par les autres objets.

La convention de définition des méthodes get et set consiste à placer `get` et `set` devant le nom de la méthode, suivi d'une espace et du nom de la propriété. Il est conseillé de commencer chaque mot suivant `get` et `set` d'une majuscule.

La variable qui stocke la valeur de propriété ne peut pas avoir le même nom que la méthode get ou set. En outre, il est d'usage de précéder le nom des variables de méthode get et set par deux traits de soulignement.

L'exemple suivant illustre la déclaration de `couleurInitiale` et les méthodes get et set utilisées pour obtenir ou définir la valeur de cette propriété.

```
...
public var __couleurInitiale:Color = 42;
...
public function get couleurInitiale():Number {
  return __couleurInitiale;
}
public function set couleurInitiale(nouvelleCouleur:Number) {
  __couleurInitiale = nouvelleCouleur;
}
```

Les lectures et les définitions sont généralement utilisées en conjonction avec des mots-clés de métadonnées afin de définir les propriétés qui soient visibles, puissent être liées ou possèdent d'autres propriétés.

Métadonnées de composant

Flash reconnaît les instructions de métadonnées de composant dans vos fichiers de classe ActionScript externes. Les balises de métadonnées peuvent définir les attributs de composant, les propriétés de liaison de données et les événements. Flash interprète ces instructions et met à jour l'environnement de développement en conséquence. Cela vous permet de définir ces membres une seule fois, plutôt que dans le code ActionScript et dans les panneaux de développement.

Les balises de métadonnées peuvent être utilisées dans les fichiers de classe ActionScript externes uniquement. Vous ne pouvez pas les utiliser dans les images d'action de vos fichiers FLA.

Utilisation de mots-clés de métadonnées

Les métadonnées sont associées à une déclaration de classe ou un champ de données individuel. Si la valeur d'un attribut est de type String, vous devez entourer cet attribut de points d'interrogation.

Les instructions de métadonnées sont liées à la ligne suivante du fichier ActionScript. Lorsque vous définissez la propriété d'un composant, ajoutez la balise des métadonnées à la ligne précédant la déclaration des propriétés. Lorsque vous définissez des événements de composant, ajoutez la balise des métadonnées hors de la définition de la classe, de sorte que l'événement soit lié à la classe entière.

Dans l'exemple suivant, les mots-clés des métadonnées Inspectable s'appliquent aux paramètres `chGout`, `chCouleur` et `chForme` :

```
[Inspectable(defaultValue="fraise")]
```

```
public var chGout:String;
[Inspectable(defaultValue="blue")]
public var chCouleur:String;
[Inspectable(defaultValue="circular")]
public var chForme:String;
```

Dans le panneau Inspecteur des propriétés et dans l'onglet Paramètres du panneau Inspecteur de composants, Flash affiche tous ces paramètres sous le type String.

Balises de métadonnées

Le tableau suivant présente les balises de métadonnées que vous pouvez utiliser dans les fichiers de classe ActionScript :

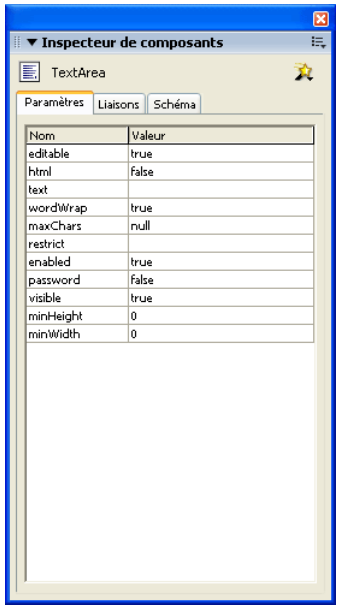
Balise	Description
Inspectable	Définit un attribut fourni aux utilisateurs de composant dans le panneau Inspecteur de composants. Pour plus d'informations, consultez *Inspectable*, page 280.
InspectableList	Identifie les sous-ensembles de paramètres d'inspection à inclure dans l'inspecteur des propriétés. Si vous n'ajoutez pas d'attribut InspectableList à votre classe de composant, tous les paramètres d'inspection s'affichent dans l'inspecteur des propriétés. Pour plus d'informations, consultez *InspectableList*, page 282.
Event	Définit les événements de composant. Pour plus d'informations, consultez *Event*, page 283.
Bindable	Révèle une propriété dans l'onglet Liaisons du panneau Inspecteur de composants. Pour plus d'informations, consultez *Bindable*, page 283.
ChangeEvent	Identifie les événements provoquant la liaison des données. Pour plus d'informations, consultez *ChangeEvent*, page 284.
ComponentTask	Désigne un ou plusieurs fichiers JSFL associés au composant qui effectue des tâches pour une occurrence de composant donnée. Pour plus d'informations, consultez *ComponentTask*, page 285.
IconFile	Le nom de fichier de l'icône représentant ce composant dans le panneau des composants Flash. Pour plus d'informations, consultez *Ajout d'une icône*, page 295.

Les sections suivantes décrivent les balises de métadonnées de composant plus en détail.

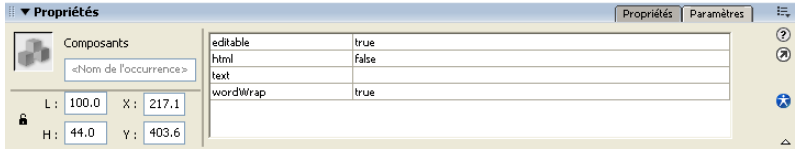
Inspectable

Les paramètres modifiables par l'utilisateur (ou « d'inspection ») d'un composant sont spécifiés dans la définition de classe du composant ; ils s'affichent dans le panneau Inspecteur de composants. Vous pouvez ainsi conserver les propriétés à inspecter et le code ActionScript sous-jacent dans un même emplacement. Pour afficher les propriétés d'un composant, faites glisser une occurrence du composant sur la scène et sélectionnez l'onglet Paramètres dans le panneau Inspecteur de composants.

La capture d'écran ci-dessous illustre l'onglet Paramètres du panneau Inspecteur de composants permettant de contrôler la zone de texte :



Vous pouvez également afficher un sous-ensemble des propriétés du composant dans l'onglet Inspecteur des propriétés, comme l'illustre l'exemple suivant :



Flash utilise le mot-clé de métadonnées Inspectable pour déterminer les paramètres à révéler dans l'environnement auteur. La syntaxe de ce mot-clé se présente comme suit :

```
[Inspectable(type_valeur=valeur[,attribut=valeur,...])]
déclaration_de_propriété nom:type;
```

L'exemple suivant définit le paramètre `enabled` comme pouvant être inspecté :

```
[Inspectable(defaultValue=true, verbose=1, category="Autre")]
var enabled:Boolean;
```

Le mot-clé Inspectable accepte également les attributs espacés, comme dans l'exemple suivant :

```
[Inspectable("danger", 1, true, maybe)]
```

L'instruction de métadonnées doit être placée juste devant la déclaration de variable de la propriété à laquelle elle doit être liée.

Le tableau suivant présente les attributs du mot-clé de métadonnées Inspectable :

Attribut	Caractères	Description
nom	String	(facultatif) Un nom d'affichage pour la propriété. Par exemple, « Largeur de police ». S'il n'est pas spécifié, utilisez le nom de la propriété, par exemple « _largeurPolice ».
type	String	(facultatif) Spécifie le type. S'il n'est pas disponible, utilisez le type de la propriété. Les valeurs suivantes sont valides : • Array • Object • List • String • Number • Boolean • Font Name • Color
defaultValue	String ou Number	(requis) Une valeur par défaut pour la propriété d'inspection.
enumeration	String	(facultatif) Spécifie une liste délimitée par des virgules de valeurs légales pour la propriété.
verbose	Number	(facultatif) Indique que cette propriété d'inspection doit être affichée uniquement si l'utilisateur spécifie que les propriétés de détail doivent être incluses. Si cet attribut n'est pas spécifié, Flash suppose que la propriété doit être affichée.
category	String	(facultatif) Groupe la propriété dans une sous-catégorie spécifique dans l'inspecteur des propriétés.
listOffset	Number	(facultatif) Ajouté pour une rétrocompatibilité avec les composants Flash MX. Utilisé comme index par défaut dans une valeur List.
variable	String	(facultatif) Ajouté pour une rétrocompatibilité avec les composants Flash MX. Utilisé pour spécifier la variable à laquelle le paramètre est lié.

InspectableList

Utilisez le mot-clé de métadonnées InspectableList pour spécifier quel sous-ensemble de paramètres d'inspection doit être affiché dans l'inspecteur des propriétés. Utilisez InspectableList de pair avec Inspectable afin de pouvoir masquer les attributs hérités pour les composants sous-classés. Si vous n'ajoutez pas de mot-clé de métadonnées InspectableList à votre classe de composant, tous les paramètres d'inspection, y compris ceux des classes parent du composant, s'affichent dans l'inspecteur des propriétés.

La syntaxe de InspectableList se présente comme suit :

```
[InspectableList("attribute1"[,...])]
// définition de classe
```

Le mot-clé InspectableList doit être placé juste devant la définition de classe parce qu'il s'applique à la classe entière.

L'exemple suivant permet d'afficher les propriétés `chGout` et `chCouleur` dans l'inspecteur des propriétés, mais exclut toutes les autres propriétés pouvant être inspectées de la classe DotParent :

```
[InspectableList("chGout","chCouleur")]
class BlackDot extends DotParent {
  [Inspectable(defaultValue="fraise")]
  public var chGout:String;
  [Inspectable(defaultValue="blue")]
  public var chCouleur:String;
  ...
}
```

Event

Utilisez le mot-clé de métadonnées Event pour définir les événements émis par ce composant.

La syntaxe de ce mot-clé se présente comme suit :

```
[Event("nom_evenement")]
```

Par exemple, le code suivant définit un événement `click` :

```
[Event("click")]
```

Ajoutez les instructions d'événement hors de la définition de classe dans le fichier ActionScript, de sorte qu'elles soient liées à la classe et non à un membre particulier de la classe.

L'exemple suivant illustre les métadonnées Evénement pour la classe UIObject, qui gère les événements `resize`, `move`, et `draw` :

```
...
import mx.events.UIEvent;
[Event("resize")]
[Event("move")]
[Event("draw")]
class mx.core.UIObject extends MovieClip {
  ...
}
```

Bindable

La liaison des données connecte les composants les uns aux autres. La liaison des données visuelles s'effectue via l'onglet Liaisons du panneau Inspecteur de composants. Cet onglet permet d'ajouter, de visualiser et de supprimer les liaisons d'un composant.

Bien que la liaison de données soit possible avec tous les composants, son objectif premier est de connecter les composants de l'interface utilisateur aux sources de données externes telles que des services Web et des documents XML. Ces sources de données sont disponibles comme composants, avec des propriétés qui peuvent être liées à d'autres propriétés de composant. Dans Flash MX Professionnel 2004, la liaison des données s'effectue principalement dans le panneau Inspecteur de composants.

Utilisez le mot-clé de métadonnées Bindable pour que les propriétés et les fonctions de lecture/définition de vos classes ActionScript s'affichent dans l'onglet Liaisons du panneau Inspecteur de composants.

La syntaxe du mot-clé de métadonnées Bindable se présente comme suit :

```
[Bindable[readonly|writeonly[,type="datatype"]]]
```

Le mot-clé Bindable doit précéder une propriété, une fonction de lecture/définition ou un autre mot-clé de métadonnées qui précède une propriété ou une fonction de lecture/définition.

L'exemple suivant définit la variable `chGout` comme une variable publique d'inspection, également accessible via l'onglet Liaisons du panneau Inspecteur de composants :

```
[Bindable]
[Inspectable(defaultValue="fraise")]
public var chGout:String = "fraise";
```

Le mot-clé de métadonnées Bindable utilise trois options spécifiant le type d'accès à la propriété, ainsi que le type de données de cette propriété. Le tableau suivant présente ces options :

Option	Description
`readonly`	Instruit Flash d'autoriser la propriété comme source d'une liaison uniquement, comme dans l'exemple suivant : `[Bindable("readonly")]`
`writeonly`	Instruit Flash d'autoriser la propriété comme destination d'une liaison uniquement, comme dans l'exemple suivant : `[Bindable("writeonly")]`
`type="`*`datatype`*`"`	Spécifie le type de données de la propriété en cours de liaison. Si vous ne spécifiez pas cette option, la liaison des données utilise le type de données de la propriété, tel qu'il est déclaré dans le code ActionScript. Si *`datatype`* est un type de données enregistré, vous pouvez utiliser la fonctionnalité dans le menu déroulant Types de données de l'onglet Schéma. L'exemple suivant définit le type de données de la propriété sur String : `[Bindable(type="String")]`

Vous pouvez combiner l'option d'accès et l'option de type de données, comme dans l'exemple suivant :

```
[Bindable(param1="writeonly",type="DataProvider")]
```

Le mot-clé Bindable est requis lorsque vous utilisez le mot-clé de métadonnées ChangeEvent. Pour plus d'informations, consultez *ChangeEvent*, page 284.

Pour plus d'informations sur la création de liaison de données dans l'environnement auteur de Flash, consultez la rubrique « Liaison des données (Flash Professionnel uniquement) » dans le guide Utilisation de Flash de l'aide.

ChangeEvent

Utilisez le mot-clé ChangeEvent pour générer un ou plusieurs événements de composant lorsque les propriétés de liaison sont modifiées.

La syntaxe de ce mot-clé se présente comme suit :

```
[Bindable]
[ChangeEvent("event"[,...)]
property_declaration or get/set function
```

A l'instar de Bindable, ce mot-clé peut être utilisé uniquement avec des déclarations de variable ou des fonctions de lecture et de définition.

Dans l'exemple suivant, le composant génère l'événement `change` lorsque la valeur de la propriété de liaison `chGout` est modifiée :

```
[Bindable]
[ChangeEvent("change")]
public var chGout:String;
```

Lorsque l'événement spécifié dans les métadonnées se produit, Flash informe les éléments liés à la propriété que cette dernière a été modifiée.

Vous pouvez également demander à votre composant de générer un événement lorsqu'une fonction de lecture ou de définition est appelée, comme dans l'exemple suivant :

```
[Bindable]
[ChangeEvent("change")]
function get selectedDate():Date
  ...
```

Dans la plupart des cas, l'événement `change` est défini sur la lecture, et distribué sur la définition.

Vous pouvez enregistrer plusieurs événements `change` dans les métadonnées, comme dans l'exemple suivant :

```
[ChangeEvent("change1", "change2", "change3")]
```

Chacun de ces événements indique une modification apportée à la variable. Tous n'ont pas besoin d'avoir lieu pour indiquer une modification.

ComponentTask

Un composant peut contenir un ou plusieurs fichiers JSFL associés qui effectuent des tâches utiles pour une occurrence de composant donnée. Ces tâches sont définies et déclarées à l'aide du mot-clé de métadonnées ComponentTask.

La syntaxe du mot-clé de métadonnées ComponentTask se présente comme suit :

```
[ComponentTask(nomTache,fichierTache,autreFichier1[,...])]
```

Le tableau suivant présente les attributs du mot-clé ComponentTask :

Attribut	Caractères	Description
nomTache	String	(requis) Le nom de la tâche, affiché comme une chaîne.
fichierTache	String	(requis) Le nom du fichier JSFL qui effectue la tâche.
autreFichier1, ...	String	(requis) Un ou plusieurs noms de fichiers requis par le fichier JSFL ; par exemple, votre tâche requiert peut-être les fichiers de description XML2UI.

Ajoutez les instructions ComponentTask hors de la définition de classe dans le fichier ActionScript, de sorte qu'elles soient liées à la classe et non à un membre particulier de la classe.

L'exemple suivant définit deux instructions ComponentTask pour la classe MonComposant :

```
[ComponentTask("Quelques tâches de
  configuration","ConfigurationDeMonComposant.jsfl","monFormulaire.xml","monA
  utreFormulaire.xml")]
[ComponentTask("Quelques tâches de
  configuration","ConfigurationDeMonAutreComposant.jsfl")]
class MonComposant {
  ...
}
```

Le menu déroulant Tâches s'affiche dans l'onglet Schéma du panneau Inspecteur de composants. Pour activer ce menu, sélectionnez une occurrence de composant sur la scène, puis cliquez sur le bouton situé à droite du panneau. Le menu contient les noms de toutes les tâches définies dans les métadonnées du composant.

Lorsque vous sélectionnez un nom de tâche dans le menu Tâches, le fichier JSFL correspondant est invoqué. Depuis le fichier JSFL, vous avez accès au composant actif, comme suit :

```
var unComposant = fl.getDocumentDOM().selection[0];
```

Lorsque vous exportez un fichier SWC, Flash inclut les fichiers JSFL associés au composant. Les fichiers JSFL et les fichiers d'aide doivent se trouver dans le même dossier que le fichier FLA lorsque vous exportez votre composant comme fichier SWC. Pour plus d'informations sur la génération de fichiers SWC, consultez *Exportation du composant*, page 293.

Définition des paramètres de composant

Lorsque vous construisez un composant, vous pouvez ajouter des paramètres pour en définir l'aspect et le comportement. Les propriétés les plus courantes s'affichent dans le panneau Inspecteur de composants comme paramètres de création. Ces propriétés sont définies à l'aide du mot-clé Inspectable (consultez *Inspectable*, page 280). Vous pouvez également définir tous les paramètres d'inspection avec ActionScript. La définition d'un paramètre avec ActionScript remplace toutes les valeurs définies en cours de programmation.

L'exemple suivant définit plusieurs paramètres de composant dans le fichier de classe Bonbon et les affiche avec le mot-clé de métadonnées Inspectable dans le panneau Inspecteur de composants :

```
class Bonbon{
  // un paramètre de chaîne
  [Inspectable(defaultValue="fraise")]
  public var chGout:String;
  // un paramètre de liste de chaînes
  [Inspectable(enumeration="acide,sucré,juteux,pourri",defaultValue="sucré")]
  public var typeGout:String;
  // un paramètre de tableau
  [Inspectable(name="Gouts", defaultValue="fraise,raisin,orange", verbose=1,
  category="Fruits")]
  var listeGouts:Array;
  // un paramètre d'objet
  [Inspectable(defaultValue="ventre:tomber,gelée:couler")]
  public var objetGelee:Object;
  // un paramètre de couleur
  [Inspectable(defaultValue="#ffffff")]
  public var couleurGelee:Color;
}
```

Les paramètres peuvent appartenir à chacun des types suivants :

- Array
- Object
- List
- String
- Number
- Boolean

- Font Name
- Color

Mise en œuvre de méthodes de base

Tous les composants doivent mettre en œuvre deux méthodes de base : taille et initialisation. Si vous ne remplacez pas ces deux méthodes dans un composant personnalisé, Flash risque de renvoyer un message d'erreur.

Mise en œuvre de la méthode d'initialisation

Flash appelle la méthode d'initialisation lorsque la classe est créée. La méthode d'initialisation doit au minimum appeler la méthode d'initialisation de la superclasse. Les paramètres de `largeur`, `hauteur` et `clip` ne sont pas correctement définis tant que cette méthode n'est pas appelée.

L'exemple suivant de méthode d'initialisation de la classe Button appelle la méthode d'initialisation de la superclasse, définit l'échelle et autres valeurs de propriétés par défaut, et obtient la valeur pour l'attribut de couleur de l'objet UIObject :

```
function init(Void):Void {
  super.init();
  labelField.selectable = false;
  labelField.styleName = this;
  useHandCursor = false;
  // marquer l'utilisation de la couleur "color"
  _color = UIObject.textColorList;
}
```

Mise en œuvre de la méthode de taille

Flash appelle la méthode de taille du composant depuis la méthode `setSize()` pour positionner le contenu du composant. La méthode de taille doit au minimum appeler la méthode de taille de la superclasse, comme dans l'exemple suivant :

```
function size(Void):Void {
  super.size();
}
```

Gestion des événements

Les événements permettent à un composant de savoir quand l'utilisateur s'est servi de l'interface et quand des modifications importantes se produisent dans l'apparence ou le cycle de vie d'un composant, telles que sa création, sa destruction ou son redimensionnement.

Le modèle d'événement est un modèle dispatcher/écouteur basé sur la spécification d'événements XML. Vous rédigez du code qui enregistre les écouteurs avec l'objet cible, de sorte que lorsque ce dernier distribue un événement, les écouteurs sont appelés.

Les écouteurs sont des fonctions ou des objets, pas des méthodes. L'écouteur reçoit un seul objet d'événement en guise de paramètre contenant le nom de l'événement et toutes les informations spécifiques à l'événement.

Les composants génèrent et distribuent les événements et consument (écoutent) d'autres événements. Lorsqu'un objet veut connaître les événements d'un autre objet, il s'enregistre auprès de ce dernier. Lors d'un événement, l'objet distribue l'événement à tous les écouteurs enregistrés en appelant une fonction requise lors de l'enregistrement. Vous devez vous enregistrer pour chaque événement afin de recevoir plusieurs événements d'un même objet.

Flash MX 2004 étend le gestionnaire d'ActionScript `on()` afin de supporter les événements de composant. Tout composant déclarant les événements dans son fichier de classe et mettant en œuvre la méthode `addEventListener()` est supporté.

Evénements courants

La liste suivante présente les événements courants diffusés par diverses classes. Chaque composant doit essayer de diffuser ces événements s'ils sont pertinents. Cette liste n'est pas exhaustive, elle recense uniquement les événements susceptibles d'être réutilisés par d'autres composants. Même si certains événements ne spécifient aucun paramètre, ils possèdent tous un paramètre implicite, à savoir une référence à l'objet diffusant l'événement.

Evénement	Paramètres	Utilisation
`click`	Aucun	Utilisé par Button ou chaque fois qu'un clic de souris n'a pas d'autre signification.
`scroll`	`Scrollbar.lineUp`, `lineDown`, `pageUp`, `pageDown`, `thumbTrack`, `thumbPosition`, `endScroll`, `toTop`, `toBottom`, `lineLeft`, `lineRight`, `pageLeft`, `pageRight`, `toLeft`, `toRight`	Utilisé par ScrollBar et par d'autres commandes activant le défilement (« butoirs » de défilement dans un menu déroulant).
`change`	Aucun	Utilisé par List, ComboBox et d'autres composants de saisie de texte.
`maxChars`	Aucun	Utilisé lorsque l'utilisateur essaie d'entrer trop de caractères dans les composants de saisie de texte.

En outre, en raison de l'héritage de UIComponent, tous les composants diffusent les événements suivants :

Evénement UIComponent	Description
`load`	Le composant crée ou charge ses sous-objets.
`unload`	Le composant décharge ses sous-objets.
`focusIn`	Le composant a le focus de saisie. Certains composants équivalents HTML (ListBox, ComboBox, Button, Text) peuvent également émettre un focus, mais tous émettent DOMFocusIn
`focusOut`	Le composant a perdu le focus de saisie.

Evénement UIComponent	Description
`move`	Le composant a été déplacé vers un nouvel emplacement.
`resize`	Le composant a été redimensionné.

Le tableau suivant présente les événements de touche courants :

Evénements de touche	Description
`keyDown`	Une touche a été enfoncée. La propriété `code` contient le code de touche et la propriété `ascii` contient le code ASCII de la touche enfoncée. Ne vérifiez pas l'objet Key de bas niveau ; il se peut que l'objet Key n'ait pas généré l'événement.
`keyUp`	Une touche a été relâchée.

Utilisation de l'objet événement

Un objet événement est transmis à un écouteur en tant que paramètre. L'objet événement est un objet ActionScript dont les propriétés contiennent les informations spécifiques à l'événement produit. Vous pouvez l'utiliser dans la fonction de rappel de l'écouteur pour déterminer le nom de l'événement diffusé ou le nom d'occurrence du composant qui a diffusé l'événement.

Par exemple, le code suivant utilise la propriété `target` de l'objet événement `evtObj` pour accéder à la propriété `label` de l'occurrence `monBouton` et pour en tracer la valeur :

```
listener = new Object();
listener.click = function(objEvt){
  trace("On a cliqué sur le bouton " + objEvt.target.label);
}
monBouton.addEventListener("click", listener);
```

Le tableau suivant présente les propriétés communes à chaque objet événement :

Propriété	Description
`type`	Une chaîne indiquant le nom de l'événement. Cette propriété est requise.
`cible`	Une référence à l'occurrence de composant qui diffuse l'événement. En règle générale, vous n'avez pas besoin de décrire cet objet de référence explicitement.

Les événements les plus courants, tels que `click` et `change`, n'ont d'autre propriété requise que `type`.

Vous pouvez construire un objet événement explicitement avant de distribuer l'événement, comme dans l'exemple suivant :

```
var objEvt = new Object();
objEvt.type = "monEvenement";
objEvt.target = this;
dispatchEvent(objEvt);
```

Vous pouvez également utiliser une syntaxe raccourcie qui définit la valeur de la propriété `type` et distribue l'événement sur une seule ligne :

```
ancestorSlide.dispatchEvent({type:"revealChild", target:this});
```

Dans l'exemple précédent, la propriété `target` n'a pas besoin d'être définie, puisqu'elle est implicite.

La description de chaque événement dans la documentation de Flash MX 2004 indique quelles propriétés d'événements sont facultatives et lesquelles sont requises. Par exemple, l'événement `ScrollBar.scroll` prend une propriété `detail` en plus des propriétés `type` et `target`. Pour plus d'informations, consultez les descriptions d'événements dans le Chapitre 4, *Dictionnaire des composants*, page 47.

Distribution d'événements

Dans le corps du fichier de classe ActionScript de votre composant, vous diffusez des événements à l'aide de la méthode `dispatchEvent()`. La signature de la méthode `dispatchEvent()` se présente comme suit :

```
dispatchEvent(objEvt)
```

Le paramètre `objEvt` est l'objet événement qui décrit l'événement (consultez *Utilisation de l'objet événement*, page 289).

Identification des gestionnaires d'événement

L'objet ou la fonction de gestionnaire d'événement qui écoute les événements de votre composant est défini dans le fichier ActionScript de votre application.

L'exemple suivant crée un objet d'écoute, gère un événement `click` et ajoute ce dernier comme écouteur d'événement à monBouton :

```
listener = new Object();
listener.click = function(objEvt){
  trace("On a cliqué sur le bouton " + objEvt.target.label);
}
monBouton.addEventListener("click", listener);
```

Hormis un objet d'écoute, vous pouvez utiliser une fonction en tant qu'écouteur. Un écouteur est une fonction si elle n'appartient pas à un objet. Par exemple, le code suivant crée la fonction écouteur `monGestionnaire` et l'enregistre sur monBouton :

```
function monGestionnaire(objEvnt){
  if (objEvnt.type == "click"){
    // votre code ici
  }
}
monBouton.addEventListener("click", monGestionnaire);
```

Pour plus d'informations sur l'utilisation de la méthode `addEventListener()`, consultez *Utilisation des écouteurs d'événements de composant*, page 23.

Lorsque vous savez qu'un objet particulier est le seul écouteur d'un événement, vous pouvez profiter du fait que le nouveau modèle d'événement appelle toujours une méthode sur l'occurrence de composant. Cette méthode est constituée du nom de l'événement et du mot `Handler`. Par exemple, pour gérer l'événement `click`, rédigez le code suivant :

```
occurrenceDeMonComposant.clickHandler = function(o){
  // insérez votre code ici
}
```

Dans le code ci-dessus, le mot-clé `this`, s'il est utilisé dans la fonction de rappel, a un domaine limité à occurrenceDeMonComposant.

Vous pouvez également utiliser des objets d'écoute supportant une méthode `handleEvent()`. Quel que soit le nom de l'événement, la méthode `handleEvent()` de l'objet d'écoute est appelée. Vous devez utiliser une instruction `if...else` ou `switch` pour traiter plusieurs événements, ce qui rend cette syntaxe maladroite. Par exemple, le code suivant utilise une instruction `if...else` pour traiter les événements `click` et `enter` :

```
monObjet.handleEvent = function(o){
  if (o.type == "click"){
    // votre code ici
  } else if (o.type == "enter"){
    // votre code ici
  }
}
target.addEventListener("click", monObjet);
target2.addEventListener("enter", monObjet);
```

Utilisation des métadonnées d'événement

Ajoutez des métadonnées d'événement à votre fichier de classe ActionScript pour chaque écouteur d'événement. La valeur du mot-clé Event devient le premier argument lors des appels de la méthode `addEventListener()`, comme dans l'exemple suivant :

```
[Event("click")] // déclaration d'événement
...
class FCheckBox{
  function addEventListener(eventName:String, eventHandler:Object) {
    ... //eventName est de type String
  }
}
```

Pour plus d'informations sur le mot-clé des métadonnées d'événement, consultez *Event*, page 283.

Réhabillage

Un contrôle d'interface utilisateur est entièrement composé de clips associés. Ceci signifie que tous les actifs d'un contrôle d'interface utilisateur peuvent être externes au clip de contrôle d'interface utilisateur, et être utilisés par d'autres composants. Par exemple, si votre composant à besoin de faire appel à des boutons, vous pouvez réutiliser les actifs de composants Button existants.

Le composant Button utilise un clip séparé pour représenter chacun de ses états (FalseDown, FalseUp, Disabled, Selected, etc.). Vous pouvez cependant associer vos clips personnalisés, appelés *enveloppes*, à ces états. A l'exécution, les clips nouveaux et anciens sont exportés dans le fichier SWF. Les anciens états deviennent simplement invisibles et cèdent la place aux nouveaux clips. Cette capacité à changer d'enveloppe lors de la programmation et de l'exécution s'appelle *réhabillage*.

Pour utiliser le réhabillage dans les composants, créez une variable pour chaque élément d'enveloppe et chaque liaison utilisé(e) par le composant. Un élément d'enveloppe peut ainsi être défini simplement en modifiant un paramètre dans le composant, comme dans l'exemple suivant :

```
var falseUpSkin = "monEnveloppe";
```

Le nom « monEnveloppe » est ensuite utilisé comme nom de liaison du symbole MovieClip pour afficher l'enveloppe false up.

L'exemple suivant illustre les variables d'enveloppe pour les différents états du composant Button :

```
var falseUpSkin:String  = "ButtonSkin";
var falseDownSkin:String  = "ButtonSkin";
var falseOverSkin:String = "ButtonSkin"
var falseDisabledSkin:String = "ButtonSkin";
var trueUpSkin:String = "ButtonSkin";
var trueDownSkin:String = "ButtonSkin";
var trueOverSkin:String = "ButtonSkin";
var trueDisabledSkin:String = "ButtonSkin";
var falseUpIcon:String = "";
var falseDownIcon:String = "";
var falseOverIcon:String = "";
var falseDisabledIcon:String = "";
var trueUpIcon:String = "";
var trueDownIcon:String = "";
var trueOverIcon:String = "";
var trueDisabledIcon:String = "";
```

Ajout de styles

L'ajout de styles consiste à enregistrer tous les éléments graphiques de votre composant avec une classe et de laisser cette dernière contrôler les modèles de couleur des graphiques à l'exécution. Aucun code spécial n'est requis pour supporter les styles lors de la mise en œuvre de composants. Les styles sont mis en œuvre exclusivement dans les classes et les enveloppes de base.

Pour plus d'informations sur les styles, consultez *Utilisation des styles pour personnaliser la couleur et le texte des composants*, page 27.

Accessibilité des composants

De plus en plus, le contenu d'un site Web doit être accessible aux personnes handicapées. Une personne malvoyante peut consulter le contenu visuel des applications Flash à l'aide d'un logiciel de lecture d'écran, qui fournit une description audio du matériel affiché à l'écran.

Flash MX 2004 inclut les fonctionnalités d'accessibilité suivantes :

- Navigation personnalisée du focus
- Raccourcis clavier personnalisés
- Documents composés d'écrans et environnement de programmation des écrans
- La classe Accessibility

Lorsque vous créez un document, vous pouvez inclure du code ActionScript pour permettre au composant et à un lecteur d'écran de communiquer. Ensuite, lorsque les développeurs utilisent votre composant pour construire une application dans Flash, ils utilisent le panneau Accessibilité pour configurer chaque occurrence de composant.

Ajoutez la ligne suivante au fichier FLA de votre composant, dans le calque dans lequel vous ajoutez d'autres appels ActionScript :

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

Par exemple, la ligne suivante permet d'accéder au composant MonBouton :

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

Lorsque les développeurs ajoutent le composant MonBouton à une application, ils peuvent utiliser le panneau Accessibilité pour le rendre accessible à des lecteurs d'écran.

Pour plus d'informations sur le panneau Accessibilité et les autres fonctionnalités d'accessibilité de Flash, consultez « Création de contenu accessible » dans le guide Utilisation de Flash de l'aide.

Exportation du composant

Flash MX 2004 exporte les composants comme paquets de composants (fichiers SWC). Lorsque vous distribuez un composant, il suffit de fournir le fichier SWC aux utilisateurs. Ce fichier contient tous les codes, fichiers SWC, images et métadonnées associés au composant ; les utilisateurs peuvent ainsi l'insérer facilement dans leur environnement Flash.

Cette section décrit un fichier SWC et explique comment importer et exporter des fichiers SWC dans Flash.

Présentation des fichiers SWC

Un fichier SWC est semblable à un fichier compressé (compressé et développé à l'aide d'un format d'archivage PKZip) généré par l'outil de programmation Flash.

Le tableau suivant présente le contenu d'un fichier SWC :

Fichier	Description
catalog.xml	(requis) Recense le contenu du paquet de composants et de ses composants individuels, et sert de répertoire aux autres fichiers du fichier SWC.
Code source	Si le composant est créé avec Flash MX 2004, le code source est constitué d'un ou plusieurs fichiers ActionScript contenant une déclaration de classe pour le composant. Le code source est utilisé uniquement pour vérifier le type lors du sous-classement des composants. Il n'est pas compilé par l'outil de programmation puisque le pseudo-code binaire est déjà présent dans le fichier SWF de mise en œuvre. Le code source peut contenir des définitions de classe intrinsèques ne contenant aucun corps de fonction ; ces dernières sont fournies uniquement pour la vérification du type.
Mise en œuvre des fichiers SWF	(requis) Fichiers SWF qui mettent en œuvre les composants. Un ou plusieurs composants peuvent être définis dans un seul fichier SWF. Si le composant est créé avec Flash MX 2004, un seul composant est exporté par fichier SWF.
Aperçu en direct de fichiers SWF	(facultatif) S'ils sont spécifiés, ces fichiers sont utilisés pour l'aperçu en direct dans l'outil de programmation. S'ils ne sont pas spécifiés, les fichiers SWF de mise en œuvre sont utilisés pour l'aperçu en direct. Le fichier SWF d'aperçu en direct peut être ignoré dans la plupart des classes. Il doit être inclus uniquement si l'aspect du composant dépend de données dynamiques (par exemple, un champ de saisie indiquant le résultat d'un appel de service web).
Informations de débogage	(facultatif) Un fichier SWD correspondant au fichier SWF de mise en œuvre. Le nom de fichier est toujours identique à celui du fichier SWF, seule l'extension .swd diffère. S'il est inclus dans le fichier SWC, le débogage du composant est autorisé.

Fichier	Description
Icône	(facultatif) Un fichier PNG contenant l'icône 18 x 18 de 8 bits par pixel, utilisé pour afficher un composant dans l'interface utilisateur de l'outil de programmation. Une icône par défaut s'affiche si aucune icône n'est fournie. (consultez *Ajout d'une icône*, page 295).
Inspecteur des propriétés	(facultatif) S'il est spécifié, ce fichier SWF est utilisé comme Inspecteur des propriétés personnalisé dans l'outil de programmation. S'il n'est pas spécifié, l'Inspecteur des propriétés par défaut s'affiche.

Pour afficher le contenu d'un fichier SWC, vous pouvez l'ouvrir à l'aide d'un utilitaire de compression supportant le format PKZip (y compris WinZip).

Vous pouvez également inclure d'autres fichiers dans le fichier SWC généré dans l'environnement Flash. Par exemple, vous voudrez peut-être inclure un fichier Lisez-moi ou le fichier FLA si vous voulez que les utilisateurs aient accès au code source du composant.

Plusieurs fichiers SWC étant développés dans un seul répertoire, chaque composant doit avoir un nom de fichier unique afin d'éviter tout conflit.

Utilisation de fichiers SWC

Cette section explique comment créer et importer des fichiers SWC. Vous devez donner des instructions pour importer un fichier SWC aux utilisateurs de votre composant.

Création de fichiers SWC

Flash MX 2004 et Flash MX Professionnel 2004 permettent de créer des fichiers SWC en exportant un clip sous forme de fichier SWC. Lors de la création d'un fichier SWC, Flash signale les erreurs de compilation, comme si vous testiez une application Flash.

Pour exporter un fichier SWC :

1. Sélectionnez un élément dans la bibliothèque Flash.
2. Cliquez avec le bouton droit de la souris (Windows) ou tout en appuyant sur la touche Contrôle (Macintosh) sur l'élément et sélectionnez Exporter le fichier SWC.
3. Enregistrez le fichier SWC.

Importation de fichiers SWC de composant dans Flash

Lorsque vous distribuez vos composants à d'autres développeurs, vous pouvez inclure les instructions suivantes afin de leur permettre de les installer et de les utiliser immédiatement.

Pour utiliser un fichier SWC dans l'environnement auteur de Flash :

1. Fermez l'environnement auteur de Flash.
2. Copiez le fichier SWC dans le répertoire *flash_root*/en/First Run/Components.
3. Démarrez l'environnement auteur de Flash ou rechargez le panneau Composants.

   L'icône du composant doit s'afficher dans le panneau Composants.

Simplification de l'utilisation du composant

Une fois que le composant est créé et préparé pour la mise en paquet, vous pouvez faciliter son utilisation. Cette section décrit certaines techniques permettant de faciliter l'utilisation des composants.

Ajout d'une icône

Vous pouvez ajouter une icône représentant votre composant dans le panneau Composants de l'environnement auteur Flash.

Pour ajouter une icône à un composant :

1. Créez une image mesurant 18 pixels et enregistrez-la au format PNG. L'image doit être de 8 bits, avec une transparence alpha ; le pixel supérieur gauche doit être transparent afin de supporter les masques.
2. Ajoutez la définition suivante au fichier de classe ActionScript de votre composant avant la définition de classe :

   ```
   [IconFile("nom_composant.png")]
   ```

3. Ajoutez l'image au répertoire contenant le fichier FLA. Lors de l'exportation du fichier SWC, Flash inclut l'image au niveau racine de l'archive.

Utilisation de l'aperçu en direct

La fonction Aperçu en direct, activée par défaut, permet de visualiser les composants sur la scène tels qu'ils apparaîtront dans le contenu Flash publié, avec leur taille approximative.

Vous n'avez plus besoin d'ajouter un aperçu en direct lorsque vous créez des composants à l'aide de l'architecture v2. Les fichiers SWC incluent le fichier SWF de mise en œuvre et le composant utilise le fichier SWF sur la scène Flash.

Ajout d'info-bulles

Les info-bulles s'affichent lorsqu'un utilisateur passe sa souris sur le nom ou l'icône d'un composant dans le panneau Composants dans l'environnement auteur Flash.

Pour ajouter des info-bulles à un composant, utilisez le mot-clé `tiptext` hors de la définition de classe dans le fichier de classe ActionScript du composant. Vous devez commenter ce mot-clé à l'aide d'un astérisque (*) et le précéder d'un symbole @ pour que le compilateur le reconnaisse correctement.

L'exemple suivant illustre le texte d'info-bulle pour le composant CheckBox :

```
* @tiptext Basic CheckBox component. Extends Button.
```

Recommandations en matière de conception d'un composant

Utilisez les techniques suivantes lors de la conception d'un composant :

- Minimisez la taille du fichier autant que possible.
- Généralisez la fonctionnalité de votre composant afin d'optimiser son potentiel d'utilisation.
- Utilisez le nouveau modèle d'événement plutôt que la syntaxe `on(event)`.
- Utilisez la classe Border plutôt que des éléments graphiques pour dessiner les bordures autours des objets.
- Utilisez le réhabillage basé sur des balises.
- Utilisez la propriété `nomSymbole`.
- Supposez un état initial. Les propriétés de style étant désormais sur l'objet, vous pouvez définir les paramètres initiaux des styles et des propriétés, de sorte que votre code d'initialisation doive les définir lors de la construction de l'objet uniquement si l'utilisateur remplace l'état par défaut.
- Lors de la définition du symbole, sélectionnez l'option Exporter dans la première image uniquement si cela est nécessaire. Flash charge le composant juste avant son utilisation dans l'application Flash ; si cette option est sélectionnée, Flash précharge le composant dans la première image de son parent. Il est inutile de précharger le composant dans la première image : sur le Web, le composant est chargé avant le préchargement, ce qui rend l'utilisation du preloader superflue.

INDEX

D

E

F

G

T