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 : Octobre 2003

Macromedia, Inc.
600 Townsend St.
San Francisco, CA 94103

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 17.

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*
- *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 (Flash Professionnel uniquement)*
- *Composant Alert (Flash Professionnel uniquement)*
- *Classes de liaison des données (Flash Professionnel uniquement)*
- *Composant DateField (Flash Professionnel uniquement)*
- *Composant DataGrid (Flash Professionnel uniquement)*
- *Composant DataHolder (Flash Professionnel uniquement)*
- *Composant DataSet (Flash Professionnel uniquement)*
- *Composant DateChooser (Flash Professionnel uniquement)*
- *Classe Form (Flash Professionnel uniquement)*
- *Composants de support (Flash Professionnel uniquement)*

- *Composant Menu (Flash Professionnel uniquement)*
- *Composant MenuBar (Flash Professionnel uniquement)*
- *Composant RDBMSResolver (Flash Professionnel uniquement)*
- *Classe Screen (Flash Professionnel uniquement)*
- *Classe Slide (Flash Professionnel uniquement)*
- *Composant Tree (Flash Professionnel uniquement)*
- *Classe WebServiceConnector (Flash Professionnel uniquement)*
- *Composant XMLConnector (Flash Professionnel uniquement)*
- *Composant XUpdateResolver (Flash Professionnel uniquement)*

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 (voir *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 :

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

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 les guides *Bien démarrer avec Flash* Utilisation de Flash, Guide de référence ActionScript et le Dictionnaire ActionScript disponibles dans l'aide.
- 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/go/flash_support_fr.

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

Pour une introduction à l'utilisation des composants, consultez le séminaire On Demand de Macromedia, Flash MX 2004 Family : Using UI Components à l'adresse : www.macromedia.com/macromedia/events/online/ondemand/index.html.

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 de la version 1 vers l'architecture de la version 2*, page 28.

Catégories de composants

Les composants inclus dans Flash MX 2004 et Flash MX Professionnel 2004 sont divisés en cinq catégories : les composants de l'interface utilisateur, les composants de données, les composants de support, les gestionnaires et les écrans. Les composants de l'interface utilisateur permettent d'interagir avec une application : par exemple, les composants RadioButton, CheckBox et TextInput sont des contrôles de l'interface utilisateur. 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 composants de support permettent de lire et de contrôler le support en flux continu ; MediaController, MediaPlayback et MediaDisplay sont les composants de support. 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. La catégorie écrans inclut les classes ActionScript qui vous permettent de contrôler les formulaires et les diapositives dans Flash MX Professionnel 2004. Pour obtenir une liste complète 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.

Les composants Flash MX 2004 et Flash MX Professionnel 2004 sont basés sur la version 2 (v2) de l'architecture des composants Macromedia. 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 de la version 1 vers l'architecture de la version 2*, page 28.

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 dans ActionScript 2.0. 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 29. Pour plus d'informations sur le traitement des événements, consultez le Chapitre 2, *Utilisation des composants*, page 17.

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 Pour plus d'informations, consultez *Composants dans le panneau Inspecteur de composants et dans l'inspecteur des propriétés*, page 18.

Le modèle d'événement écouteur permet aux objets d'écoute des fonctions de traiter les événements. Pour plus d'informations, consultez *A propos des événements de composant*, page 24.

Les propriétés des enveloppes permettent de charger des états uniquement lorsque vous en avez besoin. Pour plus d'informations, consultez *A propos de l'application des enveloppes aux composants*, page 38.

Les styles CSS permettent d'obtenir un aspect cohérent dans toutes les applications. Pour plus d'informations, consultez *Utilisation des styles pour personnaliser la couleur et le texte des composants*, page 29.

Les thèmes permettent d'appliquer un nouvel aspect à un jeu de composants. Pour plus d'informations, consultez *A propos des thèmes*, page 37.

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 27 et *Gestion de la profondeur des composants dans un document*, page 28.

Les classes de base UIObject et UIComponent fournissent les fonctionnalités élémentaires de tous les composants. Pour plus d'informations, consultez *UIComponent*, page 583 et *UIObject*, page 592.

Le format de fichier SWC permet une distribution aisée et l'utilisation d'un code dissimulable. Consultez le Chapitre 5, *Création de composants*, page 673.

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 2.0 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 le Chapitre 5, *Création de composants*, page 673 et le Guide de référence ActionScript de l'aide.

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 le Chapitre 5, *Création de composants*, page 673.

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.

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

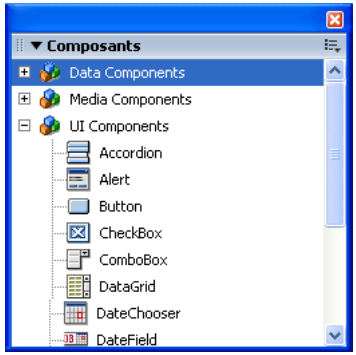
Il existe plusieurs manières d'utiliser les composants dans Macromedia Flash MX 2004 et Macromedia Flash MX Professionnel 2004. La visualisation des composants et leur ajout à un document au cours de la programmation s'effectuent dans le panneau Composants. Une fois qu'un composant a été ajouté à un 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.

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 figurant dans le dossier Macromedia\ Flash 2004\en\First Run\Components (Windows) ou Macromedia Flash 2004/en/First Run/Components (Macintosh) s'affichent dans le panneau Composants.

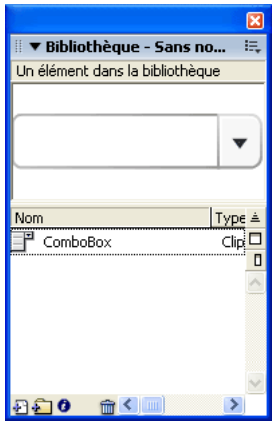
Pour afficher le panneau Composants :

- 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 (fichier 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 20.

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

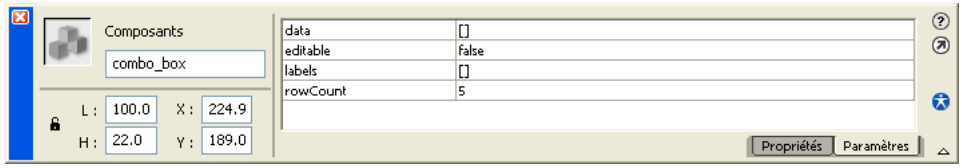
Après avoir ajouté une occurrence d'un composant dans un document Flash, utilisez l'inspecteur des propriétés pour définir et consulter des informations à son sujet. Les occurrences d'un composant sont créées par glisser-déposer sur la scène à partir du panneau Composants, puis en nommant l'occurrence 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 23.

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.

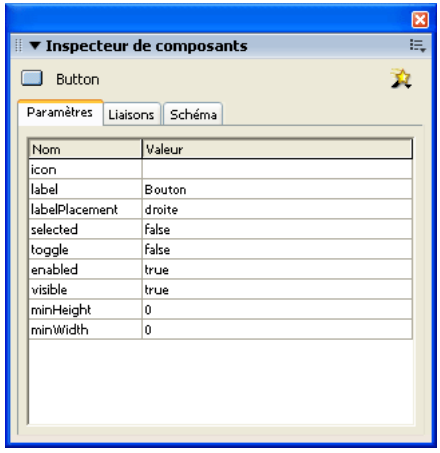


Pour consulter les paramètres d'une occurrence de composant dans le panneau Inspecteur de composants :

1 Sélectionnez Fenêtre > Panneaux de développement > Inspecteur de composants.

2 Sélectionnez une occurrence de composant sur la scène.

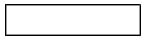
3 Pour consulter les paramètres, cliquez sur l'onglet Paramètres.



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 le Chapitre 5, *Création de composants*, page 673.

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 le Chapitre 5, *Création de composants*, page 673.

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 via l'onglet Paramètres du panneau Inspecteur de composants, 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 29.

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 l'onglet Paramètres du panneau Inspecteur de composants.

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

1. Choisissez Fenêtre > Panneaux de développement > Composants.
2. Effectuez l'une des opérations 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 23.

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 29.

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

- Appliquer un thème (voir *A propos des thèmes*, page 37).
- Modifier les enveloppes d'un composant (voir *A propos de l'application des enveloppes aux composants*, page 38).

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 (voir `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 :

```
import mx.controls.CheckBox;
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 du gestionnaire d'événements on()

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 d'événements `on()` à une occurrence de composant, de la même manière que vous affectez un gestionnaire à un bouton ou à un clip.

Lorsque vous utilisez un gestionnaire d'événements `on()`, un objet événement, `ObjEvt`, est automatiquement généré lorsque l'événement est déclenché puis transmis au gestionnaire. L'objet événement a des propriétés qui contiennent des informations sur l'événement. L'objet événement transmis au gestionnaire `on()` est systématiquement `ObjEvt`. Pour plus d'informations, consultez *UIEventDispatcher*, page 590.

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);
}
```

Pour utiliser le gestionnaire d'événements 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("l'événement " + objEvt.type + " a été diffusé");
}
```

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(objEvt){
  // 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 `objEvt` 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 *UIEventDispatcher*, page 590.

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

Remarque : Dans l'écouteur d'une fonction, le mot-clé `this` est `occurrenceDeBouton` 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, lequel doit être utilisé uniquement lorsque vous créez un composant et que vous savez qu'un objet donné est le seul écouteur d'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 le Chapitre 5, *Création de composants*, page 673.

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;
```

Vous pouvez également utiliser le panneau Accessibilité pour affecter une valeur d'indexation.

Si aucune valeur d'indexation n'est définie sur la scène, FocusManager utilise l'ordre z. L'ordre z est principalement défini par l'ordre dans lequel les composants sont placés sur la scène. Vous pouvez néanmoins utiliser les commandes Modifier/Réorganiser/Mettre au premier plan/Mettre à l'arrière-plan pour déterminer l'ordre z final.

Pour donner le focus à un composant dans une application, appelez `FocusManager.setFocus()`.

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 *Classe DepthManager*. L'API DepthManager vous permet de placer les composants de l'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 boîte de dialogue flottent au-dessus du 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 (voir *Classe DepthManager*, page 282).

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 de la version 1 vers l'architecture de la version 2

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 le Chapitre 5, *Création de composants*, page 673.

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 2.0 » 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()` (voir `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 19.

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 34.

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 31.

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 :

   ```
   nomDoccurrence.setStyle("propriété", value);
   ```

5. Choisissez Contrôle > Tester l'animation pour visualiser les modifications.

   Pour obtenir la liste des styles supportés, consultez *Styles supportés*, page 35.

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 33.

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 20.
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 35.
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 31.

Pour plus d'informations sur les propriétés de l'objet CSSStyleDeclaration, consultez *Styles supportés*, page 35. 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 20.

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 35.

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 531.

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 (voir *Classe StyleManager*, page 531). Le tableau suivant répertorie les noms de couleur par défaut :

Nom de couleur	Valeur
black	0x000000
white	0xFFFFFF
red	0xFF0000
green	0x00FF00
blue	0x0000FF
magenta	0xFF00FF
yellow	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 31, *Définition des styles pour une classe de composants*, page 32 et *Définition des styles pour des composants spécifiques*, page 31.

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 d'une bordure à trois dimensions ou section de couleur d'une bordure à deux dimensions. La valeur par défaut est 0x000000 (noir).
borderStyle	Bordure du composant : "none", "inset", "outset" ou "solid". Ce style n'hérite pas de sa valeur. La valeur par défaut est "solid".

Style	Description
buttonColor	Surface d'un bouton et section d'une bordure à trois dimensions. La valeur par défaut est 0xEFEEEF (gris clair).
color	Texte d'une étiquette de composant. La valeur par défaut est 0x000000 (noir).
disabledColor	Couleur désactivée pour du texte. La couleur par défaut est 0x848384 (gris foncé).
fontFamily	Nom de police pour du texte. La valeur par défaut est _sans.
fontSize	Taille en points pour la police. La valeur par défaut est 10.
fontStyle	Style de police : "normal" ou "italic". La valeur par défaut est "normal".
fontWeight	Epaisseur de la police : "normal" ou "bold". 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.
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 : "left", "right" ou "center". La valeur par défaut est "left".
textDecoration	Décoration du texte : "none" ou "underline" (souligné). La valeur par défaut est "none".
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 incluent 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 :

- First Run\ComponentFLA (Windows)
- First Run/ComponentFLA (Macintosh)

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. 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 29.

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, figurant dans la partie gauche de la barre d'informations au sommet 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 37.

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 37. 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 vers le bas du sous-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 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 (voir *A propos des thèmes*, page 37).

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

- Pour utiliser différentes enveloppes pour plusieurs occurrences d'un même composant, modifiez les enveloppes existantes et définissez leurs propriétés (consultez la section suivante, *Modification des enveloppes des composants*, page 40, 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 (voir *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 (voir *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, figurant dans la partie gauche de la barre d'informations au sommet 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 la section suivante, *Application d'une enveloppe modifiée à un composant*, page 40, *Application d'une enveloppe modifiée à un sous-composant*, page 41 ou *Modification des propriétés d'enveloppe dans le prototype*, page 44. Pour plus d'informations sur la manière d'appliquer une nouvelle enveloppe, consultez *A propos de l'application des enveloppes aux composants*, page 38.

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 40 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 Effectuez l'une des opérations 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 Dictionnaire ActionScript de l'aide.

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 40 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 40 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
  import 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 40 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 (Flash Professionnel uniquement)	Jeu d'affichages verticaux se chevauchant, dont les boutons supérieurs permettent aux utilisateurs de passer d'un affichage à l'autre.
Composant Alert (Flash Professionnel uniquement)	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 (true ou false).
Composant ComboBox	Permet aux utilisateurs de choisir une option dans une liste déroulante. Ce composant peut contenir un champ de texte susceptible d'être sélectionné au sommet de la liste, qui permet aux utilisateurs d'effectuer une recherche dans la liste.
Composant DateChooser (Flash Professionnel uniquement)	Permet aux utilisateurs de sélectionner une ou plusieurs dates dans un calendrier.

Composant	Description
Composant DateField (Flash Professionnel uniquement)	Champ de texte qui ne peut pas être sélectionné avec une icône de calendrier. Lorsqu'un utilisateur clique n'importe où dans le cadre de délimitation du composant, un composant DateChooser apparaît.
Composant DataGrid (Flash Professionnel uniquement)	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 (Flash Professionnel uniquement)	Permet aux utilisateurs de choisir une commande dans une liste ; menu d'application de bureau standard.
Composant MenuBar (Flash Professionnel uniquement)	Barre de menus horizontale.
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 d'effectuer une sélection parmi des options qui s'excluent réciproquement.
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 (Flash Professionnel uniquement)	Permet à un utilisateur de manipuler des informations hiérarchiques.
Composant Window	Fenêtre contenant une barre de titre, une légende, une bordure, un bouton Fermer et présentant du contenu à l'utilisateur.

Composants de données

Composant	Description
Classes de liaison des données (Flash Professionnel uniquement)	Ces classes implémentent la fonctionnalité de liaison des données Flash lors de l'exécution.
Composant DataHolder (Flash Professionnel uniquement)	Contient des données et peut être utilisé en tant que connecteur entre composants.
API DataProvider	Ce composant est le modèle des listes de données à accès linéaire. Ce modèle offre des capacités de manipulation simple de tableaux qui diffusent leurs modifications.

Composant	Description
Composant DataSet (Flash Professionnel uniquement)	Bloc de construction pour la création d'applications de données.
Composant RDBMSResolver (Flash Professionnel uniquement)	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.
Classes de service Web (Flash Professionnel uniquement)	Ces classes permettent d'accéder à des services Web qui utilisent le protocole SOAP (Simple Object Access Protocol) situé dans le paquet mx.services.
Classe WebServiceConnector (Flash Professionnel uniquement)	Fournit un accès sans script aux appels de méthode de service web.
Composant XMLConnector (Flash Professionnel uniquement)	Lit et rédige des documents XML à l'aide des méthodes HTTP GET et POST.
Composant XUpdateResolver (Flash Professionnel uniquement)	Permet d'enregistrer les données sur n'importe quelle source de données supportée. Ce composant Resolver traduit le paquet Delta au format XUpdate.

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.

Pour plus d'informations sur ces composants, consultez *Composants de support (Flash Professionnel uniquement)*, page 344.

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 Form (Flash Professionnel uniquement)	Permet de manipuler les écrans d'applications de formulaires lors de l'exécution.
Classe Screen (Flash Professionnel uniquement)	Classe de base des classes Slide et Form.
Classe Slide (Flash Professionnel uniquement)	Permet de manipuler les écrans de présentation de diapositives lors de l'exécution.

Composant Accordion (Flash Professionnel uniquement)

Le composant Accordion est un navigateur contenant une séquence d'enfants qui s'affichent un par un. Les enfants doivent être une sous-classe de la classe UIObject (qui inclut tous les composants et les écrans créés avec la version 2 de l'architecture des composants Macromedia). Cependant, le plus souvent, les enfants sont une sous-classe de la classe View. Il s'agit notamment de clips affectés à la classe mx.core.View. Pour conserver l'ordre de tabulation dans les enfants d'un accordéon, les enfants doivent également être des occurrences de la classe View.

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

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

Touche	Description
Flèche vers le bas, Flèche vers la droite	Place le focus sur l'en-tête de l'enfant suivant. Le focus englobe le premier en-tête jusqu'au dernier, sans modification de l'enfant sélectionné.
Flèche vers le haut, Flèche vers la gauche	Place le focus sur l'en-tête de l'enfant précédent. Le focus englobe le premier en-tête jusqu'au dernier, sans modification de l'enfant sélectionné.
Fin	Sélectionne le dernier enfant.
Entrée/Espace	Sélectionne l'enfant associé à l'en-tête qui a le focus.
Origine	Sélectionne le premier enfant.
Pg. Suiv.	Sélectionne l'enfant suivant. La sélection englobe le dernier enfant jusqu'au premier enfant.
Pg. Préc.	Sélectionne l'enfant précédent. La sélection englobe le premier enfant jusqu'au dernier enfant.

Touche	Description
Maj +Tab	Place le focus sur le composant précédent. Ce composant peut être situé dans l'enfant sélectionné ou hors de l'accordéon ; il ne peut en aucun cas être un autre en-tête dans le même accordéon.
Tab	Place le focus sur le composant suivant. Ce composant peut être situé dans l'enfant sélectionné ou hors de l'accordéon ; il ne peut en aucun cas être un autre en-tête dans le même accordéon.

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

Utilisation du composant Accordion (Flash Professionnel uniquement)

Le composant Accordion peut être utilisé pour présenter des formulaires comportant plusieurs parties. Par exemple, un accordéon incluant trois enfants peut présenter des formulaires dans lesquels l'utilisateur indique son adresse de livraison, son adresse de facturation, ainsi que les données de paiement correspondant à une transaction de commerce électronique. L'utilisation d'un accordéon à la place de plusieurs pages Web minimise le trafic sur le serveur et permet à l'utilisateur de bénéficier d'un meilleur sens de la progression et du contexte dans une application.

Paramètres du composant Accordion

Vous pouvez définir les paramètres de programmation suivants pour chaque occurrence de composant Accordion dans l'inspecteur des propriétés ou le panneau Inspecteur de composants :

childSymbols Tableau spécifiant les identificateurs de liaison des symboles de bibliothèque à utiliser pour créer les enfants de l'accordéon. La valeur par défaut est [] (tableau vide).

childNames Tableau spécifiant les noms d'occurrence des enfants de l'accordéon. La valeur par défaut est [] (tableau vide).

childLabels Tableau spécifiant les étiquettes de texte à utiliser sur les en-têtes de l'accordéon. La valeur par défaut est [] (tableau vide).

childIcons Tableau spécifiant les identificateurs de liaison des symboles de bibliothèque à utiliser comme icônes sur les en-têtes de l'accordéon. La valeur par défaut est [] (tableau vide).

Vous pouvez rédiger du code ActionScript pour contrôler ces paramètres et d'autres options du composant Accordion en utilisant les propriétés, méthodes et événements ActionScript. Pour plus d'informations, consultez *Classe Accordion (Flash Professionnel uniquement)*, page 55.

Création d'une application avec le composant Accordion

Dans cet exemple, un développeur d'applications conçoit la section de paiement d'un magasin en ligne. La conception appelle un accordéon avec trois formulaires, dans lesquels les utilisateurs entrent leur adresse de livraison, leur adresse de facturation et les données de paiement. Les formulaires d'adresse de livraison et d'adresse de facturation sont identiques.

Pour utiliser des écrans afin d'ajouter un composant Accordion à une application :

1 Dans Flash, sélectionnez Fichier > Nouveau, puis sélectionnez Application du formulaire Flash.

2 Double-cliquez sur le texte formulaire1, puis entrez le nom **formulaireAdresse**.

Bien que l'écran formulaireAdresse n'apparaisse pas dans la bibliothèque, il est un symbole de la classe Screen (laquelle est une sous-classe de la classe View), qu'un accordéon peut utiliser en tant qu'enfant.

3 Le formulaire étant sélectionné, dans l'inspecteur des propriétés, définissez sa propriété visible sur `false`.

Cette opération masque le contenu du formulaire dans l'application ; le formulaire apparaît uniquement dans le composant Accordion.

4 Faites glisser des composants tels que Label et TextInput du panneau Composants sur le formulaire, afin de créer un formulaire d'adresse factice ; organisez-les et définissez leurs propriétés dans le panneau Paramètres du panneau Inspecteur de composants.

Positionnez les éléments du formulaire dans le coin supérieur gauche du formulaire. Le coin supérieur gauche du formulaire est placé dans le coin supérieur gauche du composant Accordion.

5 Répétez les étapes 2 à 4 pour créer un écran appelé **formulairePaiement**.

6 Créez un nouveau formulaire appelé **formulaireAccordéon**.

7 Faites glisser un composant Accordion du panneau Composants vers le formulaire formulaireAccordéon et nommez-le **monAccordéon**.

8 Le formulaire monAccordéon étant sélectionné, dans l'inspecteur des propriétés, procédez comme suit :

- Pour la propriété childSymbols, entrez **formulaireAdresse**, **formulaireAdresse** et **formulairePaiement**.

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

  Remarque : Les deux premiers enfants sont des occurrences du même écran, car le formulaire d'adresse de livraison et le formulaire d'adresse de facturation ont des composants identiques.

- Pour la propriété childNames, entrez **adresseLivraison**, **adresseFacturation** et **paiement**.

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

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

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

9 Choisissez Contrôle > Tester l'animation.

Pour ajouter un composant Accordion à une application, procédez comme suit :

1 Choisissez Fichier > Nouveau pour créer un document Flash.

2 Sélectionnez Insertion > Nouveau symbole et appelez-le **formulaireAdresse**.

3 Dans la boîte de dialogue Créer un symbole, cliquez sur le bouton Avancé et sélectionnez Exporter pour ActionScript. Dans le champ Classe AS 2.0, entrez **mx.core.View**.

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

4 Faites glisser des composants tels que Label et TextInput du panneau Composants sur la scène afin de créer un formulaire d'adresse factice ; organisez-les et définissez leurs propriétés dans le panneau Paramètres du panneau Inspecteur de composants.

Positionnez les éléments du formulaire par rapport à 0, 0 (le milieu) sur la scène. La coordonnée 0, 0 du clip est placée dans le coin supérieur gauche du composant Accordion.

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

6 Répétez les étapes 2 à 5 pour créer un clip appelé **formulairePaiement**.

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

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

- Entrez le nom d'occurrence **monAccordéon**.
- Pour la propriété `childSymbols`, entrez **formulaireAdresse**, **formulaireAdresse** et **formulairePaiement**.

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

  Remarque : Les deux premiers enfants sont des occurrences du même clip, car le formulaire d'adresse de livraison et le formulaire d'adresse de facturation sont identiques.

- Pour la propriété `childNames`, entrez **adresseLivraison**, **adresseFacturation** et **paiement**.

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

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

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

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

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

9 Choisissez Contrôle > Tester l'animation.

Pour utiliser ActionScript afin d'ajouter des enfants à un composant Accordion, procédez comme suit :

1 Choisissez Fichier > Nouveau pour créer un document Flash.

2 Faites glisser un composant Accordion du panneau Composants jusqu'à la scène.

3 Dans l'inspecteur des propriétés, entrez **monAccordéon** comme nom d'occurrence.

4 Faites glisser un composant TextInput vers la scène et supprimez-le.

Cette opération l'ajoute à la bibliothèque afin que vous puissiez l'instancier dynamiquement à l'étape 6.

5 Dans le panneau Actions, dans l'image 1 du scénario, entrez le code suivant :

```
monAccordéon.createChild("Affichage", "adresse de livraison", { label:
  "Adresse de livraison" });
monAccordéon.createChild("Affichage", "Adresse de facturation", { label:
  "Adresse de facturation" });
monAccordéon.createChild("Affichage", "paiement", { label: "Paiement" });
```

Ce code appelle la méthode `createChild()` pour créer ses affichages enfants.

6 Dans le panneau Actions, dans l'image 1, sous le code que vous avez entré à l'étape 4, entrez le code suivant :

```
var o = monAccordéon.adresseLivraison.createChild("TextInput", "FirstName");
o.move(20, 38);
o.setSize(116, 20);
o = monAccordéon.adresseLivraison.createChild("TextInput", "lastName");
o.move(175, 38);
o.setSize(145, 20);
```

Ce code ajoute des occurrences de composant (deux composants TextInput) aux enfants de l'accordéon.

Personnalisation du composant Accordion (Flash Professionnel uniquement)

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

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

Si les en-têtes sont trop petits pour contenir leur texte d'étiquette, les étiquettes sont rognées. Si la zone de contenu d'un accordéon est plus petite qu'un enfant, l'enfant est rogné.

Utilisation de styles avec le composant Accordion

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

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 29.

Un composant Accordion gère 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.
`backgroundColor`	Couleur d'arrière-plan.
`borderColor`	Couleur de bordure.
`borderStyle`	Style de bordure ; les valeurs possibles sont "none", "solid", "inset", "outset", "default", "alert". La valeur "default" correspond à l'apparence de la bordure du composant Window, et la valeur "alert" correspond à l'apparence de la bordure du composant Alert.
`headerHeight`	Hauteur des boutons d'en-tête en pixels.
`color`	Couleur du texte d'en-tête.
`disabledColor`	Couleur d'un accordéon désactivé.

Style	Description
fontFamily	Nom de police des étiquettes d'en-tête.
fontSize	Taille en points de la police des étiquettes d'en-tête.
fontStyle	Style de police des étiquettes d'en-tête ; soit "normal", soit "italic".
fontWeight	Epaisseur de la police des étiquettes d'en-tête ; soit "normal", soit "bold".
textDecoration	Décoration du texte : "none" ou "underline".
openDuration	Durée, en millisecondes, de l'animation de transition.
openEasing	Fonction d'interpolation utilisée par l'animation.

Utilisation d'enveloppes avec le composant Accordion

Le composant Accordion utilise des enveloppes pour représenter les états visuels de ses boutons d'en-tête. Pour envelopper les boutons et la barre de titre lors de la programmation, modifiez les symboles d'enveloppes dans le dossier Flash UI Components 2/Themes/MMDefault/Accordion Assets skins states, dans la bibliothèque de l'un des fichiers FLA de thèmes. Pour plus d'informations, consultez *A propos de l'application des enveloppes aux composants*, page 38.

Un composant Accordion est constitué de sa bordure et de l'arrière-plan, de ses boutons d'en-tête et de ses enfants. Un style peut être appliqué à la bordure et à l'arrière-plan, mais aucune enveloppe ne peut être appliquée. Si aucun style ne peut être appliqué, une enveloppe peut être appliquée aux en-têtes à l'aide du sous-ensemble d'enveloppes hérité du bouton répertorié ci-dessous. Un composant Accordion utilise les propriétés d'enveloppe suivantes pour appliquer une enveloppe dynamiquement aux boutons d'en-tête :

Propriété	Description	Valeur par défaut
falseUpSkin	Etat Relevé.	accordionHeaderSkin
falseDownSkin	Etat Enfoncé.	accordionHeaderSkin
falseOverSkin	Etat Survolé.	accordionHeaderSkin
trueUpSkin	Etat Basculé.	accordionHeaderSkin

Classe Accordion (Flash Professionnel uniquement)

Héritage UIObject > UIComponent > View > Accordion

Nom de classe ActionScript mx.containers.Accordion

Un accordéon est un composant contenant des enfants qui s'affichent un par un. A chaque enfant est associé un bouton d'en-tête correspondant qui est créé lors de la création de l'enfant. Un enfant doit être une occurrence de UIObject.

Un symbole de clip devient automatiquement une occurrence de la classe UIObject lorsqu'il devient un enfant d'un accordéon. Cependant, pour conserver l'ordre de tabulation dans les enfants d'un accordéon, les enfants doivent être aussi des occurrences de la classe View. Si vous utilisez un symbole de clip comme enfant, définissez son champ Classe AS 2.0 sur `mx.core.View` afin qu'il hérite de la classe View.

La définition d'une propriété de la classe Accordion avec ActionScript annule le paramètre du même nom défini dans 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.Accordion.version);
```

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

Méthodes de la classe Accordion

Méthode	Description
`Accordion.createChild()`	Crée un enfant pour une occurrence d'un accordéon.
`Accordion.createSegment()`	Crée un enfant pour une occurrence d'un accordéon. Les paramètres de cette méthode sont différents de ceux de la méthode `createChild()`.
`Accordion.destroyChildAt()`	Détruit un enfant à la position d'index spécifiée.
`Accordion.getChildAt()`	Obtient une référence à un enfant à une position d'index spécifiée.

Hérite de toutes les méthodes des classes *UIObject*, *UIComponent* et mx.core.View.

Propriétés de la classe Accordion

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

Hérite de toutes les propriétés des classes *UIObject*, *UIComponent* et mx.core.View.

Evénements de la classe Accordion

Evénement	Description
`Accordion.change`	Diffusé à l'ensemble des écouteurs enregistrés lorsque les propriétés `selectedIndex` et `selectedChild` d'un accordéon changent suite à un clic de la souris ou à une pression sur une touche émanant de l'utilisateur.

Hérite de tous les événements des classes *UIObject*, *UIComponent* et mx.core.View.

Accordion.change

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
objetDécoute.change = function(objetEvt){
  // insérez votre code ici
}
occurrenceMonAccordéon.addEventListener("change", objetDécoute)
```

Description

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

Les composants V2 utilisent un modèle d'événement dispatcher/écouteur. Le composant Accordion distribue un événement `change` en cas de pression sur l'un de ses boutons et l'événement est géré par une fonction (également appelée *gestionnaire*) sur un objet d'écoute (*`objetDécoute`*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez une référence au gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`objetEvt`*) au gestionnaire. 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 événement, consultez *Objets événement*, page 592.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `écouteurMonAccordéon` est défini et transmis à la méthode `monAccordéon.addEventListener()` comme second paramètre. L'objet événement est capturé par le gestionnaire `change` dans le paramètre `objetEvt`. Lorsque l'événement `change` est diffusé, une instruction `trace` est envoyée au panneau Sortie, comme suit :

```
écouteurMonAccordéon = new Object();
écouteurMonAccordéon.change = function(){
  trace("remplacé par un affichage différent");
}
monAccordéon.addEventListener("change", écouteurMonAccordéon);
```

Accordion.createChild()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monAccordéon.createChild(nomClasseOuSymbole, nomOccurrence[,
  propriétésInitiales])
```

Paramètres

nomClasseOuSymbole Ce paramètre peut être la fonction constructeur de la classe de UIObject à instancier, ou le nom de liaison, référence au symbole à instancier. La classe doit être UIObject ou une sous-classe de UIObject, mais le plus souvent, il s'agit de View ou d'une sous-classe de View.

nomOccurrence Nom d'occurrence de la nouvelle occurrence.

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

- `label` Cette chaîne spécifie l'étiquette de texte que la nouvelle occurrence de l'enfant utilise sur son en-tête.
- `icon` Cette chaîne spécifie l'identificateur de liaison du symbole de bibliothèque que l'enfant utilise pour l'icône sur son en-tête.

Renvoie

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

Description

La méthode (héritée de View) crée un enfant pour le composant Accordion. L'enfant qui vient d'être créé est ajouté à la fin de la liste des enfants que possède le composant Accordion. Utilisez cette méthode pour placer des affichages dans l'accordéon. L'enfant créé est une occurrence du symbole de clip ou de classe spécifié dans le paramètre *nomClasseOuSymbole*. Vous pouvez utiliser les propriétés `label` et `icon` pour spécifier une étiquette de texte ainsi qu'une icône pour l'en-tête d'accordéon associé pour chaque enfant dans le paramètre *propriétésInitiales*.

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

Exemple

Le code suivant crée une occurrence du symbole de clip formulairePaiement nommé `paiement` en tant que dernier enfant de `monAccordéon` :

```
var enfant = monAccordéon.createChild("formulairePaiement", "paiement", {
  label: "Paiement", Icon: "icônePaiement" });
child.cardType.text = "Visa";
child.cardNumber.text = "1234567887654321";
```

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

```
var enfant = monAccordéon.createChild(mx.core.View, "paiement", { label:
  "Paiement", Icon: "icônePaiement" });
child.cardType.text = "Visa";
child.cardNumber.text = "1234567887654321";
```

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

```
import mx.core.View
var enfant = monAccordéon.createChild(View, "paiement", { label: "Paiement",
  Icon: "icônePaiement" });
child.cardType.text = "Visa";
child.cardNumber.text = "1234567887654321";
```

Accordion.createSegment()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

`monAccordéon.createSegment(`*`nomClasseOuSymbole`*`, `*`nomOccurrence`*`[, `*`label`*`[, `*`icon`*`]])`

Paramètres

`nomClasseOuSymbole` Ce paramètre peut être soit une référence à la fonction de constructeur de la classe de l'occurrence UIObject à instancier, soit le nom de liaison du symbole à instancier. La classe doit être UIObject ou une sous-classe de UIObject, mais le plus souvent, il s'agit de View ou d'une sous-classe de View.

`nomOccurrence` Nom d'occurrence de la nouvelle occurrence.

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

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

Renvoie

Référence à la nouvelle occurrence UIObject.

Description

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

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

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

Exemple

L'exemple suivant crée une occurrence du symbole de clip `formulairePaiement` nommé `paiement` en tant que dernier enfant de `monAccordéon` :

```
var enfant = monAccordéon.createSegment("FormulairePaiement", "paiement",
  "Paiement", "icônePaiement");
child.cardType.text = "Visa";
child.cardNumber.text = "1234567887654321";
```

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

```
var enfant = monAccordéon.createSegment(mx.core.View, "paiement", { label:
  "Paiement", Icon: "icônePaiement" });
child.cardType.text = "Visa";
child.cardNumber.text = "1234567887654321";
```

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

```
import mx.core.View
var enfant = monAccordéon.createSegment(View, "paiement", { label: "Paiement",
  Icon: "icônePaiement" });
child.cardType.text = "Visa";
child.cardNumber.text = "1234567887654321";
```

Accordion.destroyChildAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

`monAccordéon.destroyChildAt(`*`index`*`)`

Paramètres

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

Renvoie

Rien.

Description

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

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

Remarque : L'appel de la méthode `destroyChildAt()` diminue la propriété `numChildren` de 1.

Exemple

Le code suivant détruit le dernier enfant de `monAccordéon` :

```
monAccordéon.destroyChildAt(monAccordéon.numChildren - 1);
```

Voir aussi

`Accordion.createChild()`

Accordion.getChildAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monAccordéon.getChildAt(index)
```

Paramètres

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

Renvoie

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

Description

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

Exemple

Le code suivant obtient une référence au dernier enfant de `monAccordéon` :

```
var dernierEnfant:UIObject = monAccordéon.getChildAt(monAccordéon.numChildren
  - 1);
```

Accordion.numChildren

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monAccordéon.numChildren
```

Description

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

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

Exemple

L'exemple suivant sélectionne le dernier enfant :

```
monAccordéon.selectedIndex = monAccordéon.numChildren - 1;
```

Accordion.selectedChild

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monAccordéon.selectedChild
```

Description

Propriété : enfant sélectionné s'il existe un ou plusieurs enfants ; undefined s'il n'existe aucun enfant. Cette propriété est soit de type UIObject, soit undefined.

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

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

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

La valeur par défaut est `monAccordéon.getChildAt(0)` si l'accordéon a des enfants. Si l'accordéon n'a pas d'enfants, la valeur par défaut est undefined.

Exemple

L'exemple suivant obtient l'étiquette de l'affichage enfant sélectionné :

```
var étiquetteSélectionnée = monAccordéon.selectedChild.label;
```

L'exemple suivant obtient le formulaire de paiement qui doit être l'affichage enfant sélectionné :

```
monAccordéon.selectedChild = monAccordéon.payment;
```

Voir aussi

`Accordion.selectedIndex`

Accordion.selectedIndex

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monAccordéon.selectedIndex
```

Description

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

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

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

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

Exemple

L'exemple suivant mémorise l'index de l'enfant sélectionné :

```
var indexPrécSélectionné = monAccordéon.selectedIndex;
```

L'exemple suivant sélectionne le dernier enfant :

```
monAccordéon.selectedIndex = monAccordéon.numChildren - 1;
```

Voir aussi

`Accordion.selectedChild`, `Accordion.numChildren`

Composant Alert (Flash Professionnel uniquement)

Le composant Alert vous permet de faire apparaître une fenêtre qui présente à l'utilisateur un message et des boutons de réponse. La fenêtre Alert comprend une barre de titre dans laquelle vous pouvez insérer du texte, un message que vous pouvez personnaliser et des boutons dont les étiquettes peuvent être modifiées. Une fenêtre Alert peut inclure les boutons suivants : Oui, Non, OK et Annuler. Vous pouvez modifier les étiquettes de texte des boutons en utilisant les propriétés suivantes : `Alert.yesLabel`, `Alert.noLabel`, `Alert.okLabel` et `Alert.cancelLabel`. Vous ne pouvez pas modifier l'ordre des boutons dans une fenêtre Alert ; l'ordre des boutons est toujours OK, Oui, Non, Annuler.

Pour faire apparaître une fenêtre Alert, vous devez appeler la méthode `Alert.show()`. Pour que la méthode puisse être appelée, le composant Alert doit figurer dans la bibliothèque. Vous devez faire glisser le composant Alert du panneau Composants sur la scène, puis supprimer le composant Alert de la scène. Cette opération permet d'ajouter le composant à la bibliothèque, mais elle ne permet pas de le rendre visible dans le document.

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

Les lecteurs d'écran peuvent accéder au texte et aux boutons d'une fenêtre Alert. Lorsque vous ajoutez un composant Alert 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.AlertAccImpl.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.

Utilisation du composant Alert (Flash Professionnel uniquement)

Le composant Alert peut être utilisé chaque fois que vous voulez annoncer quelque chose à un utilisateur. Par exemple, vous pouvez faire apparaître une alerte lorsqu'un utilisateur ne remplit pas un formulaire correctement, ou lorsqu'une action atteint un certain prix, ou lorsqu'un utilisateur quitte une application sans enregistrer sa session.

Paramètres du composant Alert

Il n'existe pas de paramètres de programmation pour le composant Alert. Vous devez appeler la méthode `Alert.show()` d'ActionScript pour faire apparaître une fenêtre Alert. Vous pouvez utiliser d'autres propriétés ActionScript pour modifier la fenêtre Alert dans une application. Pour plus d'informations, consultez *Classe Alert (Flash Professionnel uniquement)*, page 66.

Création d'une application avec le composant Alert

La procédure suivante explique comment ajouter un composant Alert à une application lors de la programmation. Dans cet exemple, le composant Alert apparaît lorsqu'une action atteint un certain prix.

Pour créer une application avec le composant Alert, effectuez les opérations suivantes :

1. Double-cliquez sur le composant Alert dans le panneau Composants pour l'ajouter sur la scène.
2. Appuyez sur la touche Retour arrière (Windows) ou Supprimer (Macintosh) pour supprimer le composant de la scène.

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

```
import mx.controls.Alert
monGestionnaireClic = function (evt){
  if (evt.detail == Alert.OK){
    trace("lancer l'application boursière");
    // lancerApplicationBoursière();
  }
}
Alert.show("Lancer l'application boursière ?", "Alerte de prix d'action",
  Alert.OK | Alert.CANCEL, this, monGestionnaireClic, "icôneAction",
  Alert.OK);
```

Ce code crée une fenêtre Alert incluant les boutons OK et Annuler. En cas de pression sur l'un de ces boutons, la fonction `monGestionnaireClic` est appelée. Cependant, en cas de pression sur le bouton OK, la méthode `lancerApplicationBoursière()` est appelée.

4 Choisissez Contrôle > Tester l'animation.

Personnalisation du composant Alert (Flash Professionnel uniquement)

L'alerte se positionne elle-même au centre du composant qui a été transmis comme paramètre *parent*. Le parent doit être un composant UIComponent. S'il s'agit d'un clip, vous pouvez l'enregistrer sous mx.core.View afin qu'il hérite de UIComponent.

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

L'alerte ne répond pas à la méthode `setSize()`.

Utilisation de styles avec le composant Alert

Vous pouvez définir des propriétés de style afin de modifier l'aspect d'un composant Alert. Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, consultez *Utilisation des styles pour personnaliser la couleur et le texte des composants*, page 29.

Un composant Alert gère 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".
`buttonStyleDeclaration`	Classe (statique) CSSStyleDeclaration des styles de texte du bouton.
`messageStyleDeclaration`	Classe (statique) CSSStyleDeclaration des styles d'arrière-plan, de la bordure et du texte du message.
`titleStyleDeclaration`	Classe (statique) CSSStyleDeclaration des styles du texte du titre.

Utilisation d'enveloppes avec le composant Alert

Le composant Alert utilise des enveloppes pour représenter les états visuels de ses boutons et de sa barre de titre. Pour envelopper les boutons et la barre de titre lors de la programmation, modifiez les symboles d'enveloppes dans le dossier Flash UI Components 2/Themes/MMDefault/Window Assets skins states dans la bibliothèque de l'un des fichiers FLA de thèmes. Pour plus d'informations, consultez *A propos de l'application des enveloppes aux composants*, page 38.

Il existe du code ActionScript dans la classe RectBorder.as que le composant Alert utilise pour tracer ses bordures. Vous pouvez utiliser les styles RectBorder pour modifier un composant Alert comme suit :

```
var monAlerte = Alert.show("Ceci est un test d'erreurs", "Erreur", Alert.OK |
  Alert.CANCEL, this);
monAlerte.setStyle("Style de bordure", "incrusté");
```

Pour plus d'informations sur les styles RectBorder, consultez *Utilisation des enveloppes avec le composant List*, page 309.

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

Propriété	Description	Valeur par défaut
`buttonUp`	Etat Relevé du bouton.	`ButtonSkin`
`buttonDown`	Etat enfoncé du bouton.	`ButtonSkin`
`buttonOver`	Etat survolé du bouton.	`ButtonSkin`
`titleBackground`	Barre de titre de la fenêtre.	`TitleBackground`

Classe Alert (Flash Professionnel uniquement)

Héritage UIObject > UIComponent > View > ScrollView > Window > Alert

Nom de classe ActionScript mx.controls.Alert

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

Dans la mesure où ActionScript est asynchrone, le composant Alert n'effectue pas de blocage, ce qui signifie que les lignes de code ActionScript après l'appel de `Alert.show()` sont immédiatement exécutées. Vous devez ajouter des écouteurs pour gérer les événements `click` qui sont diffusés lorsqu'un utilisateur appuie sur un bouton, puis continuer votre code après la diffusion de l'événement.

Remarque : Dans les environnements d'exploitation qui opèrent un blocage (par exemple, Microsoft Windows), un appel de `Alert.show()` ne renvoie rien tant que l'utilisateur n'a pas effectué une action, telle qu'une pression sur un bouton.

Méthodes de la classe Alert

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

Hérite de toutes les méthodes des classes *UIObject* et *UIComponent*.

Propriétés de la classe Alert

Propriété	Description
`Alert.buttonHeight`	Hauteur de chaque bouton en pixels. La valeur par défaut est 22.
`Alert.buttonWidth`	Largeur de chaque bouton en pixels. La valeur par défaut est 100.
`Alert.cancelLabel`	Texte d'étiquette du bouton Annuler.
`Alert.noLabel`	Texte d'étiquette du bouton Non.
`Alert.okLabel`	Texte d'étiquette du bouton OK.
`Alert.yesLabel`	Texte d'étiquette du bouton Oui.

Hérite de toutes les propriétés des classes *UIObject* et *UIComponent*.

Evénements de la classe Alert

Evénement	Description
`Alert.click`	Diffusé lorsqu'une pression sur un bouton est effectuée dans une fenêtre Alert.

Hérite de tous les événements des classes *UIObject* et *UIComponent*.

Alert.buttonHeight

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
Alert.buttonHeight
```

Description

Propriété (classe) : propriété de classe (statique) qui modifie la hauteur des boutons.

Voir aussi

`Alert.buttonWidth`

Alert.buttonWidth

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
Alert.buttonWidth
```

Description

Propriété (classe) : propriété de classe (statique) qui modifie la largeur des boutons.

Voir aussi

`Alert.buttonHeight`

Alert.click

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
clickHandler = function(objetEvt){
  // insérez le code ici
}
Alert.show(message[, title[, flags[, parent[, clickHandler[, icon[,
  defaultButton]]]]]])
```

Description

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

Les composants V2 utilisent un modèle d'événement dispatcher/écouteur. Le composant Alert distribue un événement `click` en cas de pression sur l'un de ses boutons et l'événement est géré par une fonction, (également appelée *gestionnaire*), sur un objet d'écoute (*`objetDécoute`*) que vous créez. Vous appelez la méthode `Alert.show()` et lui transmettez le nom du gestionnaire en tant que paramètre. Lorsqu'une pression sur un bouton est effectuée dans la fenêtre Alert, l'écouteur est appelé.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`objetEvt`*) au gestionnaire. 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. L'objet événement de l'événement `Alert.click` a une propriété `detail` supplémentaire dont la valeur est l'une des suivantes, en fonction du bouton sur lequel l'utilisateur a cliqué : `Alert.OK`, `Alert.CANCEL`, `Alert.YES`, `Alert.NO`. Pour plus d'informations sur les objets événement, consultez *Objets événement*, page 592.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `monGestionnaireClic` est défini et transmis à la méthode `Alert.show()` comme cinquième paramètre. L'objet événement est capturé par le gestionnaire `monGestionnaireClic` dans le paramètre `evt`. La propriété `detail` de l'objet événement est ensuite utilisée dans une instruction `trace` pour envoyer le nom du bouton sur lequel l'utilisateur a appuyé (`Alert.OK` ou `Alert.CANCEL`) au panneau Sortie, comme suit :

```
monGestionnaireClic = function(evt){
  if(evt.detail == Alert.OK){
    trace(Alert.okLabel);
  }else if (evt.detail == Alert.CANCEL){
    trace(Alert.cancelLabel);
  }
}
Alert.show("Il s'agit d'un test d'erreurs", "Erreur", Alert.OK | Alert.CANCEL,
  this, monGestionnaireClic);
```

Alert.cancelLabel

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
Alert.cancelLabel
```

Description

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

Exemple

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

```
Alert.cancelLabel = "annulation";
```

Alert.noLabel

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
Alert.noLabel
```

Description

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

Exemple

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

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

Alert.okLabel

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
Alert.okLabel
```

Description

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

Exemple

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

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

Alert.show()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
Alert.show(message[, titre[, indicateurs[, parent[, gestionnaireClic[, icône[,
  boutonParDéfaut]]]]]])
```

Paramètres

`message` Message à afficher.

`titre` Texte de la barre de titre de la fenêtre Alert. Ce paramètre est facultatif. Si le paramètre *`titre`* n'est pas spécifié, la barre de titre est vide.

`indicateurs` Paramètre facultatif qui indique les boutons devant s'afficher dans la fenêtre Alert. La valeur par défaut est `Alert.OK`, ce qui permet d'afficher un bouton "OK". Lorsque vous utilisez plusieurs valeurs, insérez un caractère `|` entre chaque valeur. Il peut s'agir d'une valeur, ou de plusieurs, parmi les suivantes :

- `Alert.OK`
- `Alert.CANCEL`
- `Alert.YES`
- `Alert.NO`

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

parent Fenêtre parent du composant Alert. La fenêtre Alert se centre elle-même dans la fenêtre parent. Utilisez la valeur null ou undefined pour spécifier le scénario _root. La fenêtre parent doit hériter de la classe UIComponent. Vous pouvez enregistrer la fenêtre parent avec mx.core.View pour qu'elle hérite de la classe UIComponent. Ce paramètre est facultatif.

gestionnaireClic Gestionnaire des événements `click` diffusés lorsque l'utilisateur appuie sur les boutons. Outre les propriétés d'objet d'événement click standard, il existe une propriété `detail`, laquelle contient la valeur de l'indicateur du bouton sur lequel l'utilisateur a appuyé (`Alert.OK`, `Alert.CANCEL`, `Alert.YES`, `Alert.NO`). Ce gestionnaire peut être une fonction ou un objet. Pour plus d'informations, consultez le Chapitre 2, *Utilisation des écouteurs d'événements de composant*, page 25.

icône Chaîne correspondant à l'identificateur de liaison d'un symbole dans la bibliothèque, à utiliser comme icône s'affichant à gauche du texte. Ce paramètre est facultatif.

boutonParDéfaut Indique le bouton qui est enfoncé lorsqu'un utilisateur appuie sur Entrée (Windows) ou Retour (Macintosh). Ce paramètre peut prendre l'une des valeurs suivantes :

- `Alert.OK`
- `Alert.CANCEL`
- `Alert.YES`
- `Alert.NO`

Renvoie

L'occurrence de la classe Alert qui est créée.

Description

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

Exemple

Le code suivant est un exemple simple d'une fenêtre Alert modale contenant un bouton OK :

```
Alert.show("Bonjour à tous !");
```

Le code suivant définit un gestionnaire click qui envoie un message au panneau Sortie indiquant les boutons qui ont été enfoncés :

```
monGestionnaireClic = function(evt){
  trace (evt.detail + "a été enfoncé");
}
Alert.show("Il s'agit d'un test d'erreurs", "Erreur", Alert.OK | Alert.CANCEL,
  this, monGestionnaireClic);
```

Remarque : La propriété detail de l'objet événement renvoie un nombre permettant de représenter chaque bouton. Le bouton OK est 4, le bouton Annuler est 8, le bouton Oui est 1 et le bouton Non est 2.

Alert.yesLabel

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
Alert.yesLabel
```

Description

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

Exemple

L'exemple suivant définit l'étiquette du bouton OK sur "da" :

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

Composant Button

Le composant Button est un bouton rectangulaire de l'interface utilisateur 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 27 ou *Classe FocusManager*, page 287.

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 programmation. Cependant, dans l'aperçu en direct, une icône personnalisée est représentée sur la scène par un carré gris.

Lorsque vous ajoutez le composant Button à une application, vous pouvez utiliser le panneau Accessibilité 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.

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 Button.

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 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 false , le bouton se comporte comme un bouton-poussoir normal ; la valeur par défaut est false.

selected si le paramètre de basculement est true, ce paramètre spécifie si le bouton est enfoncé (true) ou relâché (false). 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 : left, right, top ou bottom ; la valeur par défaut est right. 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 programmation. 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 (label).
- Entrez **HelpIcon** comme paramètre de l'icône (icon).

  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 `toggle` sur true.

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 programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. Lors de l'exécution, utilisez la méthode `setSize()` (voir `UIObject.setSize()`) ou des propriétés et méthodes applicables de la classe Button (voir *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 29.

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 38 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 true, l'état du bouton ne change pas jusqu'à ce que le bouton de la souris soit relâché sur lui.

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

Les composants Button utilisent les propriétés d'enveloppe suivantes :

Propriété	Description
`falseUpSkin`	Etat Relevé. 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

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 27.

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.

Méthodes de la classe Button

Hérite de toutes les méthodes des classes *UIObject* et *UIComponent*.

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 des classes *UIObject* et *UIComponent*.

Evénements 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 des classes *UIObject* et *UIComponent*.

Button.click

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.click = function(objetEvt){
  ...
}
occurrenceDeBouton.addEventListener("click", objetDécoute)
```

Description

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

Le premier exemple d'utilisation fait appel 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 (également appelée *gestionnaire*) sur 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 transmet automatiquement un objet événement (*objetEvt*) 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()` (voir `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 événement, consultez *Objets événement*, page 592.

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 `objEvt`) 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(objEvt){
  trace("La propriété sélectionnée est passée à " + objEvt.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");
}
```

Voir aussi

`UIEventDispatcher.addEventListener()`

SimpleButton.emphasized

Disponibilité

Flash Player 6 version 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;
```

Voir aussi

SimpleButton.emphasizedStyleDeclaration

SimpleButton.emphasizedStyleDeclaration

Disponibilité

Flash Player 6 version 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.

Voir aussi

SimpleButton.emphasized

Button.icon

Disponibilité

Flash Player 6 version 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"
```

Voir aussi

`Button.labelPlacement`

Button.label

Disponibilité

Flash Player 6 version 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 `"Button"`.

Exemple

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

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

Voir aussi

`Button.labelPlacement`

Button.labelPlacement

Disponibilité

Flash Player 6 version 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 version 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
```

Voir aussi

Button.toggle

Button.toggle

Disponibilité

Flash Player 6 version 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;
```

API CellRenderer

L'API CellRenderer est un ensemble de propriétés et de méthodes que les composants basés sur des listes (List, DataGrid, Tree et Menu) utilisent pour manipuler et afficher un contenu de cellule personnalisée pour chacune de leurs lignes. Cette cellule personnalisée peut contenir un composant prédéfini, CheckBox par exemple, ou toute classe que vous créez.

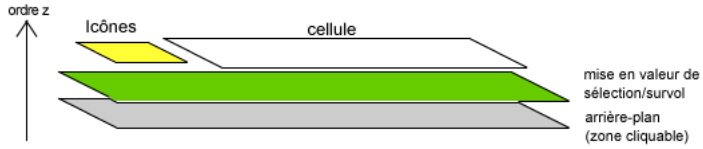
Présentation de la classe List

Pour pouvoir utiliser l'API CellRenderer, il est important de bien maîtriser la classe List. Les composants DataGrid, Tree et Menu sont des extensions de la classe List. Par conséquent, la compréhension de la classe List garantit également la compréhension de ces composants.

Remarque : Si tout composant constitue une classe, toute classe ne correspond pas nécessairement à un composant.

A propos de la composition de la classe List

Les classes List sont composées de lignes. Ces lignes affichent les mises en valeur de sélection et de survol, elles sont utilisées comme l'état « cliquable » de la sélection et jouent un rôle essentiel dans le défilement. Outre les icônes et les mises en valeur de sélection (icônes de nœud, flèches d'agrandissement d'un composant Tree, par exemple), une ligne se compose d'une cellule (ou de nombreuses cellules, dans le cas du composant DataGrid). Dans le cas de figure par défaut, ces cellules sont des objets TextField qui implémentent l'API CellRenderer. Cependant, vous pouvez indiquer à une liste d'utiliser une classe de composant différente pour chaque cellule (ligne). La seule condition requise est que la classe doit implémenter l'API CellRenderer, car la liste l'utilise pour communiquer avec la cellule.



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

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

A propos du comportement de défilement de la classe List

Les composants List utilisent un algorithme relativement complexe pour le défilement. Une liste contient le plus grand nombre de lignes qu'il est possible d'afficher simultanément ; les éléments excédant la valeur de la propriété `rowCount` n'obtiennent aucune ligne. Lors du défilement, la liste déplace toutes les lignes vers le haut ou le bas (en fonction de la direction du défilement). Elle retraite ensuite les lignes qui ne sont plus visibles ; elle les réinitialise et les utilise pour les nouvelles lignes qui défilent à l'écran, en définissant la valeur de l'ancienne ligne au nouvel élément visible et en plaçant l'ancienne ligne à l'endroit où le nouvel élément défile à l'écran.

Du fait de ce comportement de défilement, une seule cellule n'est pas utilisée pour une seule valeur. Dans la mesure où les lignes sont recyclées, le composant CellRenderer doit savoir comment réinitialiser intégralement son état lorsqu'il est défini à une nouvelle valeur. Par exemple, si votre composant CellRenderer crée une icône permettant d'afficher un élément, il peut avoir besoin de supprimer cette icône au moment d'afficher le rendu d'un nouvel élément. Supposons que votre composant CellRenderer est un conteneur qui sera rempli avec de nombreuses valeurs d'éléments au fil du temps, et qu'il doit savoir comment se modifier entièrement pour passer de l'affichage d'une valeur à l'affichage d'une autre valeur. En fait, votre cellule doit même savoir comment rendre correctement des éléments non définis, ce qui peut signifier la suppression de l'intégralité de l'ancien contenu de la cellule.

Utilisation de l'API CellRenderer

Vous devez rédiger une classe avec quatre méthodes (`CellRenderer.getPreferredHeight()`, `CellRenderer.getPreferredWidth()`, `CellRenderer.setSize()`, `CellRenderer.setValue()`) que le composant basé sur des listes utilisera pour communiquer avec la cellule.

Deux méthodes et une propriété (`CellRenderer.getCellIndex()`, `CellRenderer.getDataLabel()` et `CellRenderer.listOwner`) sont automatiquement attribuées à une cellule pour lui permettre de communiquer avec le composant basé sur des listes. Par exemple, supposons qu'une cellule comprend une case à cocher qui entraîne la sélection d'une ligne lorsque cette case est activée. Le composant CellRenderer a besoin d'une référence au composant basé sur des listes qui le contient, afin de pouvoir appeler la propriété `selectedIndex` de ce composant. en outre, la cellule doit savoir quel élément d'index elle rend actuellement afin de pouvoir définir la propriété `selectedIndex` au numéro correct ; pour ce faire, la cellule peut utiliser les méthodes `CellRenderer.listOwner` et `CellRenderer.getCellIndex()`. Il n'est pas nécessaire d'implémenter ces API ; la cellule les reçoit automatiquement lorsqu'elle est placée dans le composant basé sur des listes.

Méthodes à implémenter pour l'API CellRenderer

Vous devez rédiger une classe avec les méthodes suivantes afin que le composant List, DataGrid, Tree ou Menu puisse communiquer avec la cellule :

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

Méthodes fournies par l'API CellRenderer

Le tableau ci-dessous présente les méthodes que les composants List, DataGrid, Tree et Menu donnent à la cellule créée dans l'un de ces composants. Il n'est pas nécessaire d'implémenter ces méthodes.

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

Propriétés fournies par l'API CellRenderer

La propriété que les composants List, DataGrid, Tree et Menu donnent à la cellule créée dans l'un de ces composants est présentée ci-après. Il n'est pas nécessaire d'implémenter cette propriété.

Nom	Description
`CellRenderer.listOwner`	Référence à la liste contenant la cellule.

CellRenderer.getDataLabel()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

```
OccurrenceDeComposant.getDataLabel()
```

Paramètres

Aucun.

Renvoie

Une chaîne.

Description

Méthode : renvoie une chaîne contenant le nom du champ de données du composant CellRenderer.

Exemple

Le code suivant permet à la cellule de découvrir qu'elle rend le champ de données `"Prix"`. La variable `p` est égale à "Prix" :

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

CellRenderer.getCellIndex()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

OccurrenceDeComposant.getCellIndex()

Paramètres

Aucun.

Renvoie

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

Description

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

Exemple

Cet exemple permet de modifier la propriété dataProvider d'un composant DataGrid depuis une cellule :

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

CellRenderer.getPreferredHeight()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

OccurrenceDeComposant.getPreferredHeight()

Paramètres

Aucun.

Renvoie

La hauteur correcte de la cellule.

Description

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

Exemple

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

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

CellRenderer.getPreferredWidth()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.getPreferredWidth()

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : largeur préférée d'une cellule. Si vous indiquez une largeur plus importante que celle du composant, il est possible que la cellule soit tronquée.

Exemple

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

```
function getPreferredHeight(Void) : Number
{
   return maChaîne.length*3;
}
```

CellRenderer.listOwner

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

```
occurrenceDeComposant.listOwnner
```

Description

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

Exemple

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

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

CellRenderer.setSize()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

```
occurrenceDeComposant.setSize(largeur, hauteur)
```

Paramètres

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

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

Renvoie

Rien.

Description

Méthode : permet à la liste d'indiquer à ses cellules quelle taille elles doivent prendre. L'objet CellRenderer doit être positionné de façon à rester dans les limites de la zone décrite : dans le cas contraire, le contenu de la cellule risque de déborder dans d'autres parties de la liste et d'apparaître tronqué.

Exemple

Cet exemple dimensionne une image dans la cellule de façon à ce qu'elle reste dans les limites spécifiées par la liste :

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

CellRenderer.setValue()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.setValue(*suggérée*, *élément*, *sélectionnée*)

Paramètres

suggérée Valeur à utiliser pour le texte du composant CellRenderer, le cas échéant.

élément Objet correspondant à l'intégralité de l'élément devant être rendu. Le composant CellRenderer peut utiliser les propriétés de l'objet qu'il souhaite pour le rendu.

sélectionnée Valeur booléenne indiquant si la ligne sur laquelle figure la cellule est sélectionnée (`true`) ou non (`false`).

Renvoie

Rien.

Description

Méthode : prend les valeurs données et crée une représentation de celles-ci dans la cellule. Ceci annule les différences dans ce qui était affiché dans la cellule et ce qui doit être affiché dans la cellule du nouvel élément. Il est important de garder en mémoire qu'une cellule peut afficher de nombreuses valeurs le temps qu'elle reste dans la liste. Il s'agit de la méthode la plus importante pour les composants CellRenderer.

Exemple

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

```
function définirValeur(suggéré, élément, sélectionné) : Void
{
   //supprimer le loader
   loader.contentPath = undefined;
   // le data provider de la liste contient des URL pour différentes images
   if (suggested!=undefined)
   loader.contentPath = suggéré;
}
```

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 27 ou *Classe FocusManager*, page 287.

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.

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 : left, right, top ou bottom ; la valeur par défaut est right. 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 **LimiterAge** 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 :

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

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 545.

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 programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. Lors de l'exécution, utilisez la méthode `setSize()` (`UIObject.setSize()`) ou les autres propriétés et méthodes de la classe CheckBox (voir *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 29.

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 38.

Les composants CheckBox utilisent les propriétés d'enveloppe suivantes :

Propriété	Description
`falseUpSkin`	Etat Relevé. 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

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 27.

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);`.

Méthodes de la classe CheckBox

Hérite de toutes les méthodes des classes *UIObject* et *UIComponent*.

Propriétés de 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 des classes *UIObject* et *UIComponent*.

Evénements de 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 des classes *UIObject* et *UIComponent*.

CheckBox.click

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

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

Description

Evénement : diffusé 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 (également appelée *gestionnaire*) sur 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 transmet automatiquement un objet événement (*objetEvt*) 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 événement, consultez *Objets événement*, page 592.

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 `objEvt`) 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(objEvt){
  trace("La propriété sélectionnée est passée à " + objEvt.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");
}
```

Voir aussi

`UIEventDispatcher.addEventListener()`

CheckBox.label

Disponibilité

Flash Player 6 version 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)
```

Voir aussi

CheckBox.labelPlacement

CheckBox.labelPlacement

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

occurrenceDeCaseACocher.labelPlacement

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 93.

Exemple

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

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

Voir aussi

`CheckBox.label`

CheckBox.selected

Disponibilité

Flash Player 6 version 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 (voir *Composant TextInput*, page 545), 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, si elle est ouverte dans les versions autonome et navigateur de Flash Player.
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 renvoie le focus à la liste déroulante en mode de test d'animation.
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 déroulante se ferme dans les versions autonome et navigateur de Flash Player.
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.
Echap	Si la liste déroulante est ouverte, le focus retourne à la zone de texte et la liste déroulante se ferme en mode de test d'animation.
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 27 ou *Classe FocusManager*, page 287.

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.

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 présente 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 **IDF.swf**, **AQU.swf** et **RA.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);
}
ComboBox.addEventListener("change", form);
```

La dernière ligne de code ajoute un gestionnaire d'événements `change` à l'occurrence `ComboBox`. 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 programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation.

Si le texte est trop long pour tenir dans la liste déroulante, il est 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 29.

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. Pour plus d'informations, consultez *Utilisation de styles avec le composant Button*, page 74.
- Le texte est une occurrence TextInput et en utilise les styles. Pour plus d'informations, consultez *Utilisation des styles avec le composant TextInput*, page 547.
- La liste est une occurrence List et en utilise les styles. Pour plus d'informations, consultez *Utilisation des styles avec le composant List*, page 308.

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".

Style	Description
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 sine in/out. Pour d'autres équations, téléchargez une liste sur le site web de Robert Penner, à l'adresse www.robertpenner.com/easing/.

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 38.

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

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 des classes *UIObject* et *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.

Propriété	Description
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.
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 des classes *UIObject* et *UIComponent*.

Evénements de la classe ComboBox

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

Hérite de tous les événements des classes *UIObject* et *UIComponent*.

ComboBox.addItem()

Disponibilité

Flash Player 6 version 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 version 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 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

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

Description

Evénement : diffusé 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 (dans ce cas, `change`) qui est géré par une fonction (également appelée *gestionnaire*) sur 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 transmet automatiquement un objet événement (*objetEvt*) 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 événement, consultez *Objets événement*, page 592.

Exemple

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

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

Voir aussi

`UIEventDispatcher.addEventListener()`

ComboBox.close()

Disponibilité

Flash Player 6 version 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();
```

Voir aussi

ComboBox.open()

ComboBox.close

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

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

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque la liste déroulante commence à se fermer.

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 (dans ce cas, `close`) qui est géré par une fonction (également appelée *gestionnaire*) sur 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 transmet automatiquement un objet événement (*objetEvt*) 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 événement, consultez *Objets événement*, page 592.

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);
```

Voir aussi

`UIEventDispatcher.addEventListener()`

ComboBox.dataProvider

Disponibilité

Flash Player 6 version 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 implémentant 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 implémentant 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;

for (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 version 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.

Voir aussi

`ComboBox.dropdownWidth`

ComboBox.dropdownWidth

Disponibilité

Flash Player 6 version 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;
```

Voir aussi

`ComboBox.dropdown`

ComboBox.editable

Disponibilité

Flash Player 6 version 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 nouvelle valeur dans 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 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

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

Description

Evénement : diffusé 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 (dans ce cas, `enter`) qui est géré par une fonction (également appelée *gestionnaire*) sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

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);
```

Voir aussi

`UIEventDispatcher.addEventListener()`

ComboBox.getItemAt()

Disponibilité

Flash Player 6 version 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 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.itemRollOut = function(objetEvt){
  // 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 : diffusé 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 (dans ce cas, `itemRollOut`) qui est géré par une fonction (également appelée *gestionnaire*) sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

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

Voir aussi

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

ComboBox.itemRollOver

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.itemRollOver = function(objetEvt){
  // 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 : diffusé 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 (dans ce cas, `itemRollOver`) qui est géré par une fonction (également appelée *gestionnaire*) sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

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

Voir aussi

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

ComboBox.labelField

Disponibilité

Flash Player 6 version 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";
```

Voir aussi

List.labelFunction

ComboBox.labelFunction

Disponibilité

Flash Player 6 version 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);
}
```

Voir aussi

`List.labelField`

ComboBox.length

Disponibilité

Flash Player 6 version 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 version 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();
```

Voir aussi

`ComboBox.close()`

ComboBox.open

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

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

Description

Evénement : diffusé 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 (dans ce cas, open) qui est géré par une fonction (également appelée *gestionnaire*) sur 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 transmet automatiquement un objet événement (*objetEvt*) 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 événement, consultez *Objets événement*, page 592.

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);
```

Voir aussi

ComboBox.close, UIEventDispatcher.addEventListener()

ComboBox.removeAll()

Disponibilité

Flash Player 6 version 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();
```

Voir aussi

ComboBox.removeItemAt(), ComboBox.replaceItemAt()

ComboBox.removeItemAt()

Disponibilité

Flash Player 6 version 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);
```

Voir aussi

ComboBox.removeAll(), ComboBox.replaceItemAt()

ComboBox.replaceItemAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

OccurrenceDeListeDéroulante.replaceItemAt(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. 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");
```

Voir aussi

`ComboBox.removeAll()`, `ComboBox.removeItemAt()`

ComboBox.rowCount

Disponibilité

Flash Player 6 version 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 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.scroll = function(objetEvt){
  // 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 : diffusé 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 (dans ce cas, `scroll`) qui est géré par une fonction (également appelée *gestionnaire*) sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

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

Voir aussi

`UIEventDispatcher.addEventListener()`

ComboBox.selectedIndex

Disponibilité

Flash Player 6 version 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;
```

Voir aussi

`ComboBox.selectedItem`

ComboBox.selectedItem

Disponibilité

Flash Player 6 version 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);
```

Voir aussi

`ComboBox.dataProvider`, `ComboBox.selectedIndex`

ComboBox.text

Disponibilité

Flash Player 6 version 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 version 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 version 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);
```

Classes de liaison des données (Flash Professionnel uniquement)

Les classes de liaison des données fournissent les fonctionnalités d'exécution de la fonction de liaison des données dans Flash MX Professionnel 2004. Vous pouvez créer et configurer visuellement des liaisons de données dans l'environnement de programmation Flash via l'onglet Liaisons du panneau Inspecteur de composants. Vous pouvez également créer et configurer par programmation des liaisons à l'aide des classes du paquet mx.data.binding.

Pour un aperçu de la liaison des données et pour savoir comment créer visuellement des liaisons de données dans l'outil de programmation Flash, consultez « Liaison des données (Flash Professionnel uniquement) » dans le guide Utilisation de Flash de l'aide.

Disponibilité des classes de liaison des données lors de l'exécution (Flash Professionnel uniquement)

Pour que les classes du service de liaison des données puissent être utilisées lors de l'exécution, le composant DataBindingClasses doit figurer dans la bibliothèque de votre fichier FLA. Lorsque vous créez visuellement des liaisons dans l'environnement de programmation Flash, ce composant est automatiquement ajouté dans la bibliothèque de votre document. Cependant, si vous utilisez uniquement ActionScript pour créer des liaisons lors de l'exécution, vous devez alors ajouter ce composant manuellement dans la bibliothèque de votre document. Pour plus d'informations sur l'ajout de ce composant à votre document, consultez « Utilisation de la liaison des données et des services web à l'exécution (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Classes dans le paquet mx.data.binding (Flash Professionnel uniquement)

Le tableau suivant répertorie les classes dans le paquet mx.data.binding.

Classe	Description
Classe Binding (Flash Professionnel uniquement)	Crée une liaison entre deux points de fin.
Classe ComponentMixins (Flash Professionnel uniquement)	Ajoute une fonctionnalité spécifique à la liaison aux composants.
Classe CustomFormatter (Flash Professionnel uniquement)	Classe de base pour la création de classes de mise en forme personnalisées.
Classe CustomValidator (Flash Professionnel uniquement)	Classe de base pour la création de classes de validation personnalisées.
Classe DataType (Flash Professionnel uniquement)	Offre un accès en lecture et en écriture aux champs de données d'une propriété d'un composant.

Classe	Description
Classe EndPoint (Flash Professionnel uniquement)	Définit la source ou la destination d'une liaison.
Classe TypedValue (Flash Professionnel uniquement)	Contient une valeur de données et des informations concernant le type de données de la valeur.

Classe Binding (Flash Professionnel uniquement)

Nom de classe ActionScript mx.data.binding.Binding

La classe Binding définit une association entre deux points de fin, une *source* et une *destination*. Elle recherche les modifications apportées au point de fin source et copie les données modifiées vers le point de fin de destination chaque fois que la source change.

Vous pouvez rédiger des liaisons personnalisées à l'aide de la classe Binding (et des classes de prise en charge), ou vous pouvez utiliser l'onglet Liaisons dans le panneau Inspecteur de composants (Fenêtre > Panneaux de développement > Inspecteur de composants).

Remarque : Pour rendre cette classe disponible lors de l'exécution, vous devez inclure le composant DataBindingClasses dans votre document FLA. Pour plus d'informations, consultez « Utilisation de la liaison des données et des services web à l'exécution (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Pour un aperçu des classes dans le paquet mx.data.binding, consultez *Classes de liaison des données (Flash Professionnel uniquement)*, page 128.

Méthodes de la classe Binding

Méthode	Description
`Binding.execute()`	Extrait les données du composant source, les met en forme et les affecte au composant de destination.

Constructeur de la classe Binding

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

```
new binding(source, destination, [format], [estBidirectionnel])
```

Paramètres

source Point de fin source de la liaison. Ce paramètre est de type `mx.data.binding.EndPoint`, mais il peut être tout objet ActionScript doté des champs de point de fin requis (consultez *Classe EndPoint (Flash Professionnel uniquement)*).

destination Point de fin de destination de la liaison. Ce paramètre est de type `mx.data.binding.EndPoint`, mais il peut être tout objet ActionScript doté des champs de point de fin requis (consultez *Classe EndPoint (Flash Professionnel uniquement)*).

format (Facultatif) Objet qui contient des informations de format. Les propriétés de l'objet doivent être les suivantes :

- `cls` Classe ActionScript qui étend la classe `mx.data.binding.DataAccessor`.
- `settings` Objet dont les propriétés fournissent des paramètres facultatifs pour la classe de mise en forme spécifiée par `cls`.

estBidirectionnel (Facultatif) Valeur booléenne qui spécifie si le nouvel objet Binding est bidirectionnel (`true`) ou non (`false`). La valeur par défaut est `false`.

Renvoie

Rien.

Description

Constructeur : crée un nouvel objet Binding. Vous pouvez lier des données à tout objet ActionScript qui a des propriétés et émet des événements, entre autres, des composants.

Un objet de liaison existe tant que le clip principal contient les composants source et de destination. Par exemple, si un clip nommé "A" contient les composants "X" et "Y" et qu'il existe une liaison entre "X" et "Y", la liaison est effective tant que le clip A existe.

Remarque : Il n'est pas nécessaire de conserver une référence au nouvel objet Binding. Dès que l'objet Binding est créé, il commence immédiatement à rechercher les événements « modifiés » émis par l'un des points de fin. Toutefois, dans certains cas, vous pouvez opter pour l'enregistrement d'une référence du nouvel objet Binding, afin de pouvoir appeler sa méthode `execute()` ultérieurement (consultez `Binding.execute()`).

Exemple

Exemple 1 : Dans cet exemple, la propriété `text` d'un composant TextInput (`src_txt`) est liée à la propriété `text` d'un autre composant TextInput (`dest_txt`). Lorsque le champ de texte `src_txt` perd le focus (c'est-à-dire, lorsque l'événement `focusOut` est généré), la valeur de sa propriété `text` est copiée dans `dest_txt.text`.

```
import mx.data.binding.*;
var src = new EndPoint();
src.component = src_txt;
src.property = "text";
src.event = "focusOut";

var dest= new EndPoint();
dest.component = dest_txt;
dest.property = "text";

new Binding(src, dest);
```

Exemple 2 : Cet exemple explique comment créer un objet Binding qui utilise une classe de mise en forme personnalisée. Pour plus d'informations sur la création de classes de mise en forme personnalisées, consultez *Classe CustomFormatter (Flash Professionnel uniquement)*, page 132.

```
import mx.data.binding.*;
var src = new EndPoint();
src.component = src_txt;
src.property = "text";
src.event = "focusOut";

var dest= new EndPoint();
dest.component = text_dest;
```

```
dest.property = "text";

new Binding(src, dest, {cls: mx.data.formatters.Custom, settings: {classname:
  "com.masociété.SpecialFormatter"}});
```

Binding.execute()

Disponibilité

Flash Player 6.

Edition

Flash MX Professionnel 2004.

Utilisation

maLiaison.execute([*inverser*])

Paramètres

inverser Valeur booléenne qui indique si la liaison doit aussi être exécutée de la destination vers la source (`true`), ou uniquement de la source vers la destination (`false`). Par défaut, cette valeur est définie sur `false`.

Renvoie

Une valeur `null` si la liaison est exécutée correctement ; dans le cas contraire, renvoie un tableau de messages d'erreur (chaînes) qui décrivent l'erreur, ou les erreurs, qui ont empêché l'exécution de la liaison.

Description

Méthode : recherche les données du composant source et les affecte au composant de destination. Si la liaison utilise une mise en forme, les données sont mises en forme avant leur affectation à la destination.

Cette méthode valide également les données et entraîne l'émission d'un événement `valid` ou `invalid` par les composants source et de destination. Les données sont affectées à l'événement de destination même s'il n'est pas valide, sauf si la destination est en lecture seule.

Si le paramètre *inverser* est défini sur `true`, ***et*** si la liaison est bidirectionnelle, la liaison est alors exécutée dans l'ordre inverse (de la destination vers la source).

Exemple

Le code suivant, associé à une occurrence d'un composant Button, exécute la liaison dans l'ordre inverse (du composant de destination vers le composant source) lorsque l'utilisateur clique sur le bouton.

```
on(click) {
  _root.maLiaison.execute(true);
}
```

Classe CustomFormatter (Flash Professionnel uniquement)

Nom de classe ActionScript mx.data.binding.CustomFormatter

La classe CustomFormatter définit deux méthodes, `format()` et `unformat()`, qui permettent de transformer des valeurs de données d'un type spécifique en type String, et inversement. Par défaut, ces méthodes n'ont aucun rôle ; vous devez les implémenter dans une sous-classe de mx.data.binding.CustomFormatter.

Pour créer votre propre mise en forme personnalisée, vous devez créer une sous-classe de CustomFormatter qui implémente les méthodes `format()` et `unformat()`. Vous pouvez ensuite affecter cette classe à une liaison entre des composants, soit en créant un nouvel objet Binding avec ActionScript (consultez *Classe Binding (Flash Professionnel uniquement)*, page 129), soit en utilisant l'onglet Liaisons dans le panneau Inspecteur de composants. Pour plus d'informations sur l'affectation d'une classe de mise en forme à l'aide de l'inspecteur de composants, consultez « Mises en forme de schéma (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Vous pouvez également affecter une classe de mise en forme à une propriété de composant dans l'onglet Schéma du panneau Inspecteur de composants. Toutefois, dans ce cas, la mise en forme sera utilisée uniquement lorsque les données doivent être sous la forme d'une chaîne. A l'inverse, les mises en forme affectées via le panneau Liaisons, ou créées avec ActionScript, sont utilisées chaque fois que la liaison est exécutée.

Pour obtenir un exemple de l'écriture et de l'affectation d'une mise en forme personnalisée à l'aide d'ActionScript, consultez *Exemple de mise en forme personnalisée*, page 132.

Remarque : Pour rendre cette classe disponible lors de l'exécution, vous devez inclure le composant DataBindingClasses dans votre document FLA. Pour plus d'informations, consultez « Utilisation de la liaison des données et des services web à l'exécution (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Pour un aperçu des classes dans le paquet mx.data.binding, consultez *Classes de liaison des données (Flash Professionnel uniquement)*, page 128.

Exemple de mise en forme personnalisée

L'exemple suivant explique comment créer une classe de mise en forme personnalisée, avant de l'appliquer à une liaison entre deux composants à l'aide d'ActionScript. Dans cet exemple, la valeur actuelle d'un composant NumericStepper (sa propriété `value`) est liée à la valeur actuelle d'un composant TextInput (sa propriété `text`). La classe de mise en forme personnalisée met en forme la valeur numérique actuelle du composant NumericStepper (par exemple, 1, 2 ou 3) en utilisant le terme anglais équivalent (par exemple, "one", "two" ou "three") avant de l'affecter au composant TextInput.

Pour créer et utiliser une mise en forme personnalisée :

1 Dans Flash MX Professionnel 2004, créez un nouveau fichier ActionScript.

2 Ajoutez le code suivant au fichier :

```
// NumberFormatter.as
class NumberFormatter extends mx.data.binding.CustomFormatter {
  // Met en forme un nombre, renvoie une chaîne
  function format(rawValue) {
    var returnValue;
    var strArray = new Array('one', 'two', 'three');
    var numArray = new Array(1, 2, 3);
```

```
        returnValue = 0;
        for (var i = 0; i<strArray.length; i++) {
          if (rawValue == numArray[i]) {
            returnValue = strArray[i];
            break
          }
        }
        return returnValue;
      } // convertit une valeur mise en forme, renvoie une valeur brute
      function unformat(formattedValue) {
        var returnValue;
        var strArray = new Array('one', 'two', 'three');
        var numArray = new Array(1, 2, 3);
        returnValue = "non valide";
        for (var i = 0; i<strArray.length; i++) {
          if (formattedValue == strArray[i]) {
            returnValue = numArray[i];
            break
          }
        }
        return returnValue;
      }
    }
```

3 Enregistrez le fichier ActionScript sous NumberFormatter.as.

4 Créez un nouveau document Flash (FLA).

5 Ouvrez le panneau Composants (Fenêtre > Panneaux de développement > Composants).

6 Faites glisser un composant TextInput sur la scène et nommez-le **textInput**.

7 Faites glisser un composant NumericStepper sur la scène et nommez-le **stepper**.

8 Ouvrez le scénario (Fenêtre > Scénario) et sélectionnez la première image du calque 1.

9 Ouvrez le panneau Actions (Fenêtre > Panneaux de développement > Actions).

10 Ajoutez le code suivant au panneau Actions

```
import mx.data.binding.*;
var x:NumberFormatter;
var liaisonPersonnalisée = new Binding({component:stepper, property:"value",
  event:"change"}, {component:textInput, property:"text",
  event:"enter,change"}, {cls:mx.data.formatters.Custom,
  settings:{classname:"NumberFormatter"}});
```

La seconde ligne de code (`var x:NumberFormatter`) garantit que le code binaire de votre classe de mise en forme personnalisée est inclus dans le fichier SWF compilé.

11 Sélectionnez Fenêtre > Autres panneaux > Bibliothèques communes > Classes pour ouvrir la bibliothèque Classes.

12 Ouvrez votre bibliothèque de document en sélectionnant Fenêtre > Bibliothèque.

13 Faites glisser le composant DataBindingClasses de la bibliothèque Classes dans votre bibliothèque de document.

Grâce à cette opération, les classes d'exécution de liaison des données sont disponibles pour votre document. Pour plus d'informations, consultez « Utilisation de la liaison des données et des services web à l'exécution (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

14 Enregistrez le fichier FLA dans le dossier qui contient NumberFormatter.as.

15 Testez le fichier (Contrôle > Tester l'animation).

Cliquez sur les boutons du composant NumericStepper et observez le contenu de la mise à jour du composant TextInput.

Méthodes de la classe CustomFormatter

Méthode	Description
`CustomFormatter.format()`	Convertit un type de données brut en chaîne de texte.
`CustomFormatter.unformat()`	Convertit une chaîne de texte en type de données brut.

CustomFormatter.format()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

Cette méthode est appelée automatiquement ; vous ne l'invoquez pas directement.

Paramètres

`donnéesBrutes` Données à mettre en forme.

Renvoie

Valeur mise en forme.

Description

Méthode : convertit un type de données brut en un nouvel objet.

Cette méthode n'est pas implémentée par défaut. Vous devez la définir dans votre sous-classe de mx.data.binding.CustomFormatter.

Exemple

Pour plus d'informations, consultez *Exemple de mise en forme personnalisée*, page 132.

CustomFormatter.unformat()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

Cette méthode est appelée automatiquement ; vous ne l'invoquez pas directement.

Paramètres

`donnéesMisesEnForme` Données mises en forme à reconvertir en type de données brut.

Renvoie

Valeur non mise en forme.

Description

Méthode : convertit une chaîne, ou un autre type de données, en type de données brut. Cette transformation doit réaliser exactement la transformation inverse de `CustomFormatter.format()`.

Cette méthode n'est pas implémentée par défaut. Vous devez la définir dans votre sous-classe de mx.data.binding.CustomFormatter.

Pour plus d'informations, consultez *Exemple de mise en forme personnalisée*, page 132.

Classe CustomValidator (Flash Professionnel uniquement)

Nom de classe ActionScript mx.data.binding.CustomValidator

Utilisez la classe CustomValidator pour effectuer une validation personnalisée d'un champ de données contenu dans un composant.

Pour créer une classe de validation personnalisée, vous créez tout d'abord une sous-classe de mx.data.binding.CustomValidator qui implémente une méthode appelée `validate()`. Une valeur est automatiquement transmise à cette méthode afin qu'elle soit validée. Pour plus d'informations sur l'implémentation de cette méthode, consultez `CustomValidator.validate()`.

Ensuite, vous affectez votre classe de validation personnalisée à un champ d'un composant via l'onglet Schéma du panneau Inspecteur de composants. Pour obtenir un exemple de création et d'utilisation d'une classe de validation personnalisée, consultez la section Exemple dans la description de `CustomValidator.validate()`.

Pour affecter une classe de validation personnalisée, procédez comme suit :

1. Dans le panneau Inspecteur de composants (Fenêtre > Inspecteur de composants), sélectionnez l'onglet Schéma.
2. Sélectionnez le champ à valider, puis sélectionnez Personnalisé dans le menu déroulant Types de données.
3. Sélectionnez le champ Options de validation (au bas de l'onglet Schéma), puis cliquez sur l'icône en forme de loupe afin d'ouvrir la boîte de dialogue Paramètres de validation personnalisés.
4. Dans la zone de texte Classe ActionScript, entrez le nom de la classe de validation personnalisée que vous avez créée.

   Pour que la classe que vous spécifiez soit incluse dans le fichier SWF publié, elle doit figurer dans le chemin de classe.

Remarque : Pour rendre cette classe disponible lors de l'exécution, vous devez inclure le composant DataBindingClasses dans votre document FLA. Pour plus d'informations, consultez « Utilisation de la liaison des données et des services web à l'exécution (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Pour un aperçu des classes dans le paquet mx.data.binding, consultez *Classes de liaison des données (Flash Professionnel uniquement)*, page 128.

Méthodes de la classe CustomValidator

Méthode	Description
`CustomValidator.validate()`	Effectue la validation des données.
`CustomValidator.validationError()`	Signale les erreurs de validation.

CustomValidator.validate()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

Cette méthode est appelée automatiquement ; vous ne l'invoquez pas directement.

Paramètres

`valeur` Données à valider ; cette valeur peut être de tout type.

Renvoie

Rien.

Description

Méthode : appelée automatiquement pour valider les données contenues dans le paramètre *`valeur`*. Vous devez implémenter cette méthode dans votre sous-classe de CustomValidator ; l'implémentation par défaut n'a aucune incidence.

Vous pouvez utiliser tout code ActionScript de votre choix pour examiner et valider les données. Si les données ne sont pas valides, cette méthode doit appeler `this.validationError()` avec un message d'erreur approprié. Vous pouvez appeler `this.validationError()` plusieurs fois s'il existe plusieurs problèmes de validation au niveau des données.

Dans la mesure où la méthode `validate()` peut être appelée à plusieurs reprises, vous devez éviter d'ajouter du code à cette méthode dont l'exécution est longue. Votre implémentation de cette méthode doit uniquement vérifier que les données sont valides puis signaler les erreurs à l'aide de `CustomValidator.validationError()`. De même, votre implémentation ne doit prendre aucune mesure suite au test de validation (par exemple, alerter l'utilisateur final). En fait, vous devez créer des écouteurs d'événements pour les événements `valid` et `invalid`, puis alerter l'utilisateur final à partir de ces écouteurs d'événements (consultez l'exemple suivant).

Exemple

La procédure ci-après explique comment créer et utiliser une classe de validation personnalisée. La méthode `validate()` de la classe CustomValidator, NombresImpairs.as, détermine comme non valide toute valeur qui n'est pas un nombre impair. La validation a lieu lorsque la valeur d'un composant NumericStepper change, en relation avec la propriété `text` d'un composant Label.

Pour créer et utiliser une classe de validation personnalisée :

1 Dans Flash MX Professionnel 2004, créez un nouveau fichier ActionScript (AS).

2 Ajoutez le code suivant dans le fichier AS :

```
class NombresImpairs extends mx.data.binding.CustomValidator
{
  public function validate(value) {
    // vérification que la valeur est un nombre
    var n = Number(value);
    if (String(n) == "NaN") {
      this.validationError("'" + value + "' n'est pas un nombre.");
      return;
    }
    // vérification que le nombre est impair
    if (n % 2 == 0) {
      this.validationError("'" + value + "' n'est pas un nombre impair.");
      return;
    }
    // les données sont OK, aucune action requise, renvoi uniquement
  }
}
```

3 Enregistrez le fichier AS sous NombresImpairs.as.

Remarque : Le nom du fichier AS doit correspondre au nom de la classe.

4 Créez un nouveau document Flash (FLA).

5 Ouvrez le panneau Composants (Fenêtre > Panneaux de développement > Composants).

6 Faites glisser un composant NumericStepper du panneau Composants sur la scène et appelez-le **stepper**.

7 Faites glisser un composant Label sur la scène et appelez-le **textLabel**.

8 Faites glisser un composant TextArea sur la scène et appelez-le **status**.

9 Sélectionnez le composant NumericStepper, puis ouvrez le panneau Inspecteur de composants (Fenêtre > Panneaux de développement > Inspecteur de composants).

10 Sélectionnez l'onglet Liaisons dans le panneau Inspecteur de composants, puis cliquez sur le bouton Ajouter une liaison (+).

11 Sélectionnez la propriété `Value` (la seule) dans la boîte de dialogue Ajouter une liaison, puis cliquez sur OK.

12 Dans le panneau Inspecteur de composants, double-cliquez sur bound to dans le panneau des attributs de liaison de l'onglet Liaisons pour ouvrir la boîte de dialogue Lié à.

13 Dans la boîte de dialogue Lié à, sélectionnez le composant Label dans le panneau Chemin du composant, ainsi que sa propriété `text` dans le panneau Emplacement du schéma. Cliquez sur OK.

14 Sélectionnez le composant Label sur la scène et cliquez sur l'onglet Schéma dans le panneau Inspecteur de composants.

15 Dans le panneau des attributs du schéma, sélectionnez Personnalisé dans le menu déroulant Types de données.

16 Double-cliquez sur le champ Options de validation dans le panneau des attributs du schéma pour ouvrir la boîte de dialogue Paramètres de validation personnalisés.

17 Dans la zone de texte Classe ActionScript, entrez **NombresImpairs**, qui est le nom de la classe ActionScript que vous avez créée préalablement. Cliquez sur OK.

18 Ouvrez le scénario (Fenêtre > Scénario) et sélectionnez la première image du calque 1.

19 Ouvrez le panneau Actions (Fenêtre > Actions).

20 Ajoutez le code suivant au panneau Actions

```
function dataIsInvalid(evt) {
  if (evt.property == "text") {
    status.text = evt.messages;
  }
}

function dataIsValid(evt) {
  if (evt.property == "text") {
    status.text = "OK";
  }
}

textLabel.addEventListener("valid", dataIsValid);
textLabel.addEventListener("invalid", dataIsInvalid);
```

21 Enregistrez le fichier FLA sous OddOnly.fla dans le dossier qui contient NombresImpairs.as.

22 Testez le fichier SWF (Contrôle > Tester l'animation).

Cliquez sur les flèches du composant NumericStepper afin de modifier sa valeur. Vous pouvez remarquer qu'un message apparaît dans le composant TextArea lorsque vous choisissez des nombres pairs et impairs.

CustomValidator.validationError()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`this.validationError(`*`messageErreur`*`)`

Remarque : Cette méthode peut être invoquée uniquement depuis une classe de validation personnalisée ; le mot clé `this` fait référence à l'objet CustomValidator actif.

Paramètres

`messageErreur` Chaîne qui contient le message d'erreur à signaler.

Renvoie

Rien.

Description

Méthode : vous appelez cette méthode depuis la méthode `validate()` de votre sous-classe de CustomValidator pour signaler les erreurs de validation. Si vous n'appelez pas cette méthode, un événement `valid` est généré à la fin de `validate()`. Si vous appelez cette méthode une ou plusieurs fois depuis la méthode `validate()`, un événement `invalid` est généré après le renvoi de `validate()`.

Chaque message que vous transmettez à `validationError()` est disponible dans la propriété "messages" de l'objet événement transmis au gestionnaire d'événements `invalid`.

Exemple

Consultez la section Exemple de `CustomValidator.validate()`.

Classe EndPoint (Flash Professionnel uniquement)

Nom de classe ActionScript mx.data.binding.EndPoint

La classe EndPoint définit la source ou la destination d'une liaison. Les objets EndPoint définissent une valeur constante, une propriété de composant ou un champ particulier d'une propriété de composant, à partir desquels vous pouvez obtenir des données, ou auxquels vous pouvez affecter des données. Ils peuvent également définir un événement ou une liste d'événements qu'un objet Binding attend ; lorsque l'événement spécifié survient, la liaison est exécutée.

Lorsque vous créez une nouvelle liaison avec le constructeur de classe Binding, vous la transmettez à deux objets EndPoint : un pour la source et un pour la destination.

```
new mx.data.binding.Binding(srcEndPoint, destEndPoint);
```

Les objets EndPoint, `srcEndPoint` et `destEndPoint`, peuvent être définis comme suit :

```
var srcEndPoint = new mx.data.binding.EndPoint();
var destEndPoint = new mx.data.binding.EndPoint();
srcEndPoint.component = source_txt;
srcEndPoint.property = "text";
srcEndPoint.event = "focusOut";
destEndPoint.component = dest_txt;
destEndPoint.property = "text";
```

Le code précédent signifie « lorsque le champ de texte source perd le focus, copiez la valeur de sa propriété `text` dans la propriété `text` du champ de texte de destination ».

Vous pouvez également transmettre des objets ActionScript génériques au constructeur Binding, au lieu de transmettre explicitement des objets EndPoint construits. La seule condition requise est que les objets définissent les propriétés EndPoint nécessaires, à savoir `component` et `property`. Le code suivant est équivalent au code présenté précédemment.

```
var srcEndPoint = {component:source_txt, property:"text"};
var destEndPoint = {component:dest_txt, property:"text"};
new mx.data.binding.Binding(srcEndPoint, destEndPoint);
```

Remarque : Pour rendre cette classe disponible lors de l'exécution, vous devez inclure le composant DataBindingClasses dans votre document FLA. Pour plus d'informations, consultez « Utilisation de la liaison des données et des services web à l'exécution (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Pour un aperçu des classes dans le paquet mx.data.binding, consultez *Classes de liaison des données (Flash Professionnel uniquement)*, page 128.

Propriétés de la classe EndPoint

Méthode	Description
`EndPoint.constant`	Valeur constante.
`EndPoint.component`	Référence à une occurrence de composant.
`EndPoint.property`	Nom d'une propriété de l'occurrence du composant spécifiée par `EndPoint.component`.
`EndPoint.location`	Emplacement d'un champ de données dans la propriété de l'occurrence du composant.
`EndPoint.event`	Nom d'un événement, ou d'une liste d'événements, que l'occurrence du composant émettra lorsque les données changeront.

Constructeur de la classe EndPoint

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

```
new EndPoint()
```

Renvoie

Rien.

Description

Constructeur : crée un nouvel objet EndPoint.

Exemple

Cet exemple crée un nouvel objet EndPoint appelé `source_txt` et affecte des valeurs à ses propriétés `component` et `property`.

```
var source_obj = new mx.data.binding.EndPoint();
source_obj.component = monChampDeTexte;
source_obj.property = "text";
```

EndPoint.constant

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

```
endPoint_src.constant
```

Description

Propriété : valeur constante affectée à un objet EndPoint. Cette propriété peut être affectée uniquement à des objets EndPoint qui sont la source (et non la destination) d'une liaison entre des composants. La valeur peut être tout type de données compatible avec la destination de la liaison. Si cela est précisé, toutes les autres propriétés EndPoint de l'objet EndPoint spécifié sont ignorées.

Exemple

Dans cet exemple, la valeur de chaîne constante "bonjour" est affectée à la propriété constante d'un objet EndPoint :

```
var sourceEndPoint = new mx.data.binding.EndPoint();
sourceEndPoint.constant="bonjour";
```

EndPoint.component

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

endPointObj.component

Description

Propriété : référence à une occurrence de composant.

Exemple

Cet exemple affecte une occurrence du composant List (`listBox1`) en tant que paramètre composant d'un objet EndPoint.

```
var sourceEndPoint = new mx.data.binding.EndPoint();
sourceEndPoint.component=listBox1;
```

EndPoint.property

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

endPointObj.property

Description

Propriété : spécifie un nom de propriété de l'occurrence de composant spécifiée par `EndPoint.component` qui contient les données susceptibles d'être liées.

Remarque : `EndPoint.component` et `EndPoint.property` doivent être associés pour former une combinaison objet/propriété ActionScript valide.

Exemple

Cet exemple lie la propriété `text` d'un composant TextInput (`text_1`) à la même propriété d'un autre composant TextInput (`text_2`).

```
var sourceEndPoint = {component:text_1, property:"text"};
var destEndPoint = {component:text_2, property:"text"};
new Binding(sourceEndPoint, destEndPoint);
```

EndPoint.location

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`endPointObj``.location`

Description

Propriété : spécifie l'emplacement d'un champ de données dans la propriété de l'occurrence du composant. Un emplacement peut être spécifié de quatre manières : sous la forme d'une chaîne qui contient une expression XPath ou un chemin ActionScript, sous la forme d'un tableau de chaînes ou sous la forme d'un objet.

Les expressions XPath peuvent être utilisées uniquement lorsque les données sont un objet XML. Pour obtenir la liste des expressions XPath prises en charge, consultez « Expressions XPath supportées », dans le guide Utilisation de Flash de l'aide (consultez l'exemple 1 ci-après).

Pour les objets XML et ActionScript, vous pouvez également spécifier une chaîne qui contient un chemin ActionScript. Un chemin ActionScript contient les noms de champs séparés par des points (par exemple, `"a.b.c"`).

Vous pouvez également spécifier un tableau de chaînes comme emplacement. Chaque chaîne du tableau analyse un autre niveau d'imbrication. Vous pouvez utiliser cette technique avec des données XML et ActionScript (consultez l'exemple 2 ci-dessous). Lorsqu'un tableau de chaînes est utilisé avec des données ActionScript, cela équivaut à utiliser ActionScript ; c'est-à-dire que le tableau `["a","b","c"]` est équivalent à `"a.b.c"`.

Si vous spécifiez un objet comme emplacement, l'objet doit spécifier deux propriétés : `path` et `indices`. La propriété `path` est un tableau de chaînes, comme indiqué précédemment, sauf qu'une ou plusieurs des chaînes spécifiées peuvent être le symbole spécial `"[n]"`. Pour chaque occurrence de ce symbole dans `path`, un élément d'index correspondant doit exister dans `indices`. Lorsque le chemin est évalué, les index sont utilisés pour l'indexation dans les tableaux. L'élément d'index peut être tout objet EndPoint. Ce type d'emplacement ne peut pas être appliqué aux données XML ; il est appliqué exclusivement aux données ActionScript (consultez l'exemple 3 ci-après).

Exemple

Exemple 1 : Cet exemple utilise une expression XPath pour spécifier l'emplacement d'un nœud appelé `zip` dans un objet XML.

```
var sourceEndPoint = new mx.databinding.EndPoint();
var sourcObj=new Object();
sourceObj.xml=new XML("<zip>94103</zip>");
sourceEndPoint.component=sourceObj;
sourceEndPoint.property="xml";
sourceEndPoint.location="/zip";//
```

Exemple 2 : Cet exemple utilise un tableau de chaînes pour effectuer une analyse poussée d'une propriété de clip imbriquée.

```
var sourceEndPoint = new mx.data.binding.EndPoint();
//supposons que movieClip1.ball.position existe
ssourceEndPoint.component=movieClip1;
sourceEndPoint.property="ball";
//accès à movieClip1.ball.position.x
sourceEndPoint.location=["position","x"];
```

Exemple 3 : Cet exemple indique comment utiliser un objet pour spécifier l'emplacement d'un champ de données dans une structure de données complexe.

```
var ville=new Object();
ville.cinémas = [{cinéma: "t1", films: [{name: "Le bon, la brute et le
  truand"}, {name:"Matrix Reloaded"}]}, {cinéma: "t2", films: [{name:
  "Gladiator"}, {name: "Arrête-moi si tu peux"}]}];
var srcEndPoint = new EndPoint();
srcEndPoint.component=ville;
srcEndPoint.property="cinémas";
srcEndPoint.location = {path: ["[n]","films","[n]","name"], indices:
  [{constant:0},{constant:0}]};
```

EndPoint.event

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`objEndPoint``.event`

Description

Propriété : spécifie le nom d'un événement, ou un tableau de noms d'événements, généré par le composant lorsque les données affectées à la propriété de liaison changent. Lorsque l'événement a lieu, la liaison est exécutée.

L'événement spécifié s'applique uniquement aux composants qui sont utilisés comme source d'une liaison, ou comme destination d'une liaison bidirectionnelle. Pour plus d'informations sur la création de liaisons bidirectionnelles, consultez *Classe Binding (Flash Professionnel uniquement)*, page 129.

Exemple

Dans cet exemple, la propriété `text` d'un composant TextInput (`src_txt`) est liée à la même propriété d'un autre composant TextInput (`dest_txt`). La liaison est exécutée lorsque l'événement `focusOut` ou `enter` est émis par le composant `src_txt`.

```
var source = {component:src_txt, property:"text", event:["focusOut",
  "enter"]};
var dest = {component:maZoneDeTexte, property:"text"};
var nouvelleLiaison = new mx.data.binding.Binding(source, dest);
```

Classe ComponentMixins (Flash Professionnel uniquement)

Nom de classe ActionScript mx.data.binding.ComponentMixins

La classe ComponentMixins définit les propriétés et les méthodes automatiquement ajoutées à un objet qui est la source ou la destination d'une liaison, ou à un composant qui est la cible d'un appel de la méthode `ComponentMixins.initComponent()`. Ces propriétés et méthodes n'ont aucune incidence sur les fonctionnalités du composant ; elles ajoutent des fonctionnalités utiles pour la liaison des données.

Remarque : Pour rendre cette classe disponible lors de l'exécution, vous devez inclure le composant DataBindingClasses dans votre document FLA. Pour plus d'informations, consultez « Utilisation de la liaison des données et des services web à l'exécution (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Pour un aperçu des classes dans le paquet mx.data.binding, consultez *Classes de liaison des données (Flash Professionnel uniquement)*, page 128.

Méthodes de la classe ComponentMixins

Méthode	Description
`ComponentMixins.getField()`	Renvoie un objet permettant d'obtenir et de définir la valeur d'un champ à un emplacement spécifique dans une propriété de composant.
`ComponentMixins.initComponent()`	Ajoute les méthodes ComponentMixin à un composant.
`ComponentMixins.refreshFromSources()`	Exécute toutes les liaisons qui ont ce composant comme objet Endpoint de destination.

Méthode	Description
`ComponentMixins.refreshDestinations()`	Exécute toutes les liaisons qui ont cet objet comme objet Endpoint source.
`ComponentMixins.validateProperty()`	Vérifie si les données de la propriété indiquée sont valides.

ComponentMixins.getField()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

occurrenceDeComposant.getField(*nomDeProp*, [*emplacement*])

Paramètres

nomDeProp Chaîne contenant le nom d'une propriété du composant spécifié.

emplacement (Facultatif) Emplacement d'un champ au sein de la propriété du composant. Ceci est utile si la propriété du composant spécifiée par `nomDeProp` est une structure de données complexe et si vous êtes intéressé par un champ présentant cette structure. Cette propriété peut prendre l'une des trois formes suivantes :

- Une chaîne contenant une expression XPath. Cette forme est valide uniquement pour les structures de données XML. Pour obtenir la liste des expressions XPath prises en charge, consultez « Expressions XPath supportées », dans le guide Utilisation de Flash de l'aide.
- Une chaîne contenant des noms de champs, séparés par des points ("a.b.c", par exemple). Cette forme est autorisée pour toutes les données complexes (ActionScript ou XML).
- Un tableau de chaînes dans lequel chaque chaîne est un nom de champ (["a", "b", "c"], par exemple). Cette forme est autorisée pour toutes les données complexes (ActionScript ou XML).

Renvoie

Un objet DataType.

Description

Méthode : renvoie un objet DataType dont vous pouvez utiliser les méthodes pour obtenir ou définir la valeur de données dans la propriété d'un composant à l'emplacement de champ spécifié. Pour plus d'informations, consultez *Classe DataType (Flash Professionnel uniquement)*, page 149.

Exemple

Cet exemple utilise la méthode `DataType.setAsString()` pour définir la valeur d'un champ situé dans la propriété d'un composant. Dans ce cas, la propriété (`results`) est une structure de données complexe.

```
import mx.data.binding.*;
var champ : DataType = monComposant.getField("résultats", "po.adresse.nom1");
champ.setAsString("Teri Randall");
```

Voir aussi

`DataType.setAsString()`

ComponentMixins.initComponent()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`mx.data.binding.ComponentMixins.initComponent(`*`occurrenceDeComposant`*`)`

Paramètres

`occurrenceDeComposant` Référence à une occurrence de composant.

Renvoie

Rien.

Description

Méthode (statique) : ajoute toutes les méthodes ComponentMixins au composant spécifié par *`occurrenceDeComposant`*. Cette méthode est appelée automatiquement pour tous les composants impliqués dans une liaison de données. Pour que les méthodes ComponentMixins puissent être utilisées par un composant qui n'est pas impliqué dans une liaison de données, vous devez appeler explicitement cette méthode pour le composant.

Exemple

Le code suivant met les méthodes ComponentMixins à la disposition d'un composant DataSet.

```
mx.data.binding.ComponentMixins.initComponent(_root.monEnsembleDeDonnées);
```

ComponentMixins.refreshFromSources()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`occurrenceDeComposant``.refreshSources()`

Renvoie

Rien.

Description

Méthode : exécute toutes les liaisons pour lesquelles *occurrenceDeComposant* est l'objet EndPoint de destination. Cette méthode vous permet d'exécuter des liaisons qui ont des sources constantes ou des sources n'émettant pas d'événement "data changed" (données modifiées).

Exemple

L'exemple suivant exécute toutes les liaisons pour lesquelles l'occurrence de composant ListBox appelée `listeVilles` est l'objet EndPoint de destination.

```
listeVilles.refreshFromSources();
```

ComponentMixins.refreshDestinations()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

occurrenceDeComposant`.refreshDestinations()`

Renvoie

Rien.

Description

Méthode : exécute toutes les liaisons pour lesquelles *occurrenceDeComposant* est l'objet EndPoint source. Cette méthode vous permet d'exécuter des liaisons dont les sources n'émettent pas d'événement "data changed" (données modifiées).

Exemple

L'exemple suivant exécute toutes les liaisons pour lesquelles l'occurrence de composant DataSet appelée `données_utilisateur` est l'objet EndPoint source.

```
données_utilisateur.refreshDestinations();
```

ComponentMixins.validateProperty()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

occurrenceDeComposant`.validateProperty(`*nomDeProp*`)`

Paramètres

nomDeProp Chaîne contenant le nom d'une propriété appartenant à *occurrenceDeComposant*.

Renvoie

Un tableau ou la valeur `null`.

Description

Méthode : détermine si les données contenues dans la chaîne *`nomDeProp`* sont valides en fonction des paramètres de schéma de la propriété. Les paramètres de schéma de la propriété sont ceux spécifiés sous l'onglet Schéma, dans le panneau Inspecteur de composants.

La méthode renvoie la valeur `null` si les données sont valides ; dans le cas contraire, elle renvoie un tableau de messages d'erreur au format chaîne.

La validation s'applique uniquement aux champs pour lesquels des informations de schéma sont disponibles. Si un champ est un objet contenant d'autres champs, chaque champ enfant sera validé, de manière récursive. Chaque champ distribuera un événement `valid` ou `invalid`, selon le cas. Pour chaque champ de données contenu par *`nomDeProp`*, cette fonction distribue des événements `valid` ou `invalid`, comme suit :

- Si la valeur du champ est `nulle` et *n'est pas* obligatoire, la méthode renvoie la valeur `null`. Aucun événement n'est généré.
- Si la valeur du champ est `nulle` et *est* obligatoire, une erreur est renvoyée et un événement `invalid` est généré.
- Si la valeur du champ n'est pas nulle et le schéma du champ ne possède *pas* de validateur, la méthode renvoie la valeur `null` ; aucun événement n'est généré.
- Si la valeur n'est pas nulle et le schéma du champ ne définit *pas* de validateur, les données sont alors traitées par l'objet validateur. Si les données sont valides, un événement `valid` est généré et la valeur `null` est renvoyée ; dans le cas contraire, un événement `invalid` est généré et un tableau de chaînes d'erreur est renvoyé.

Exemple

Les exemples suivants indiquent comment utiliser la méthode `validateProperty()` pour s'assurer que le texte saisi par un utilisateur est d'une longueur valide. Vous déterminez la longueur valide en définissant les paramètres de validation des chaînes (paramètre Validation Options) du type de données String, sous l'onglet Schéma du panneau Inspecteur de composants. Si l'utilisateur saisit une chaîne d'une longueur non valide dans un champ de texte, les messages d'erreur renvoyés par la méthode `validateProperty()` s'affichent dans le panneau de sortie.

Pour valider le texte saisi par un utilisateur dans un composant TextInput :

1. Faites glisser un composant TextInput du panneau Composants (Fenêtre > Panneaux de développement > Composants) sur la scène, puis nommez-le **codePostal_txt**.
2. Sélectionnez le composant TextInput, puis dans le panneau Inspecteur de composants (Fenêtre > Panneaux de développement > Composants), cliquez sur l'onglet Schéma.
3. Dans le panneau de l'arborescence de schéma (panneau supérieur de l'onglet Schéma), sélectionnez la propriété `text`.
4. Dans le panneau des attributs du schéma (panneau inférieur de l'onglet Schéma), sélectionnez ZipCode dans le menu déroulant du paramètre Data Type.
5. Choisissez Fenêtre > Scénario pour ouvrir le scénario, s'il n'est pas déjà ouvert.
6. Cliquez sur la première image du calque 1 dans le scénario, puis ouvrez le panneau Actions (Fenêtre > Actions).

7 Ajoutez le code suivant au panneau Actions

```
// Ajouter les méthodes ComponentMixin au composant TextInput.
// Cette étape est requise uniquement si le composant
// n'est pas déjà impliqué dans une liaison de données,
// en tant que source ou destination.
mx.data.binding.ComponentMixins.initComponent(codePostal_txt);
// Définition de la fonction d'écouteur d'événements pour le composant :
validationRésultats = function (objEvt) {
  var erreurs:Array = objEvt.target.validateProperty("text");
  if (errors != null) {
    trace(erreurs);
  }
};
// Enregistrement de la fonction d'écouteur avec le composant :
codePostal_txt.addEventListener("enter", validationRésultats);
```

8 Choisissez Fenêtre > Autres panneaux > Bibliothèques communes > Classes pour ouvrir la bibliothèque Classes.

9 Ouvrez la bibliothèque de votre document en choisissant Fenêtre > Bibliothèque.

10 Faites glisser le composant DataBindingClasses de la bibliothèque Classes dans le panneau de la bibliothèque de votre document.

Cette étape est nécessaire pour que le fichier SWF puisse utiliser les classes d'exécution de liaison des données au moment de son exécution. Pour plus d'informations, consultez « Utilisation de la liaison des données et des services web à l'exécution (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

11 Testez le fichier SWF en choisissant Contrôle > Tester l'animation.

Dans le composant TextInput, sur la scène, entrez un code postal américain non valide ; par exemple, un code qui contient des lettres ou qui contient moins de cinq chiffres. Vérifiez les messages d'erreur dans le panneau de sortie.

Classe DataType (Flash Professionnel uniquement)

Nom de classe ActionScript mx.data.binding.DataType

La classe DataType offre un accès en lecture et en écriture aux champs de données d'une propriété de composant. Pour obtenir un objet DataType, vous devez appeler la fonction `ComponentMixins.getField()` sur un composant. Vous pouvez ensuite appeler les méthodes de l'objet DataType pour obtenir et définir la valeur du champ.

La différence entre l'obtention et la définition de valeurs de champs à l'aide des méthodes de l'objet DataType et l'obtention ou la définition de ces mêmes valeurs directement dans l'occurrence du composant est que, dans le deuxième cas, ces données sont fournies sous leur forme « brute ». Inversement, lorsque vous obtenez ou définissez des valeurs de champs à l'aide des méthodes de la classe DataType, ces valeurs sont traitées selon les paramètres de schéma du champ.

Par exemple, le code suivant obtient directement la valeur de la propriété d'un composant puis l'affecte à une variable. La variable, `VarProp`, contient la valeur « brute » correspondant à la valeur active de la propriété `nomPropriété`.

```
var VarProp = monComposant.nomPropriété;
```

L'exemple suivant obtient la valeur de cette même propriété à l'aide de la méthode `DataType.getAsString()`. Dans ce cas, la valeur affectée à `VarChaîne` est la valeur de `nomPropriété` suite à son traitement selon ses paramètres de schéma puis à son renvoi sous forme de chaîne.

```
var objDataType:mx.data.binding.DataType = monComposant.getField("nomProp");
var VarChaîne : Chaîne = objDataType.getAsString();
```

Pour plus d'informations sur la méthode de spécification des paramètres de schéma d'un champ, consultez « Utilisation des schémas dans l'onglet Schéma (Flash Professionnel uniquement) », dans le manuel Utilisation de Flash de l'aide.

Vous pouvez également utiliser les méthodes de la classe DataType pour obtenir ou définir des champs dans différents types de données. La classe DataType convertit automatiquement les données brutes dans le type demandé, si cela est possible. Par exemple, dans l'exemple de code précédent, les données extraites sont converties au format String (Chaîne), même si les données brutes sont d'un type différent.

La méthode `ComponentMixins.getField()` peut être utilisée pour les composants inclus dans une liaison de données (en tant que source, destination ou index) ou ayant été initialisés à l'aide de la méthode `ComponentMixins.initComponent()`. Pour plus d'informations, consultez *Classe ComponentMixins (Flash Professionnel uniquement)*, page 144.

Remarque : Pour rendre cette classe disponible lors de l'exécution, vous devez inclure le composant DataBindingClasses dans votre document FLA. Pour plus d'informations, consultez « Utilisation de la liaison des données et des services web à l'exécution (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Pour un aperçu des classes dans le paquet mx.data.binding, consultez *Classes de liaison des données (Flash Professionnel uniquement)*, page 128.

Méthodes de la classe DataType

Méthode	Description
`DataType.getAsBoolean()`	Extrait la valeur active du champ sous forme de valeur booléenne (Boolean).
`DataType.getAsNumber()`	Extrait la valeur active du champ sous forme de nombre (Number).
`DataType.getAsString()`	Extrait la valeur active du champ sous forme de chaîne (String).
`DataType.getAnyTypedValue()`	Extrait la valeur active du champ.
`DataType.getTypedValue()`	Extrait la valeur active du champ sous la forme du type de données demandé.
`DataType.setAnyTypedValue()`	Définit une nouvelle valeur dans le champ.
`DataType.setAsBoolean()`	Définit le champ à la nouvelle valeur, donnée sous forme booléenne.
`DataType.setAsNumber()`	Définit le champ à la nouvelle valeur, donnée sous forme de nombre.
`DataType.setAsString()`	Définit le champ à la nouvelle valeur, donnée sous forme de chaîne.
`DataType.setTypedValue()`	Définit une nouvelle valeur dans le champ.

Propriétés de la classe DataType

Propriété	Description
DataType.encoder	Fournit une référence à l'objet Encoder associé à ce champ.
DataType.formatter	Fournit une référence à l'objet Formatter associé à ce champ.
DataType.kind	Fournit une référence à l'objet Kind associé à ce champ.

DataType.encoder

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

objetDataType.encoder

Description

Propriété : fournit une référence à l'objet Encoder associé à ce champ, s'il existe. Vous pouvez utiliser cette propriété pour accéder aux propriétés et aux méthodes définies par l'encodeur spécifique appliqué au champ dans l'onglet Schéma du panneau Inspecteur de composants.

Si aucun encodeur n'a été appliqué au champ concerné, la propriété renvoie la valeur `undefined`.

Pour plus d'informations sur les encodeurs inclus dans Flash MX Professionnel 2004, consultez « Encodeurs de schéma (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Exemple

Dans l'exemple suivant, nous supposons que le champ auquel l'utilisateur accède (`valide`) utilise l'encodeur Boolean (`mx.data.encoders.Bool`). Cet encodeur est inclus dans Flash MX Professionnel 2004 et contient une propriété appelée `trueStrings`, qui spécifie les chaînes devant être interprétées en tant que valeurs booléennes `true`. Le code suivant définit la propriété `trueStrings` de l'encodeur d'un champ afin qu'elle corresponde aux chaînes "yes" et "oui".

```
var monChamp:mx.data.binding.DataType = dataSet.getField("valide");
monChamp.encoder.trueStrings = "Yes,Oui";
```

DataType.formatter

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

objectDataType.formatter

Description

Propriété : fournit une référence à l'objet de mise en forme (Formatter) associé à ce champ, s'il existe. Vous pouvez utiliser cette propriété pour accéder aux propriétés et méthodes de l'objet de mise en forme appliqué au champ, dans l'onglet Schéma du panneau Inspecteur de composants.

Si aucun objet de mise en forme n'a été appliqué au champ en question, la propriété renvoie la valeur `undefined`.

Pour plus d'informations sur les encodeurs inclus dans Flash MX Professionnel 2004, consultez « Mises en forme de schéma (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Exemple

Dans cet exemple, nous supposons que le champ auquel l'utilisateur accède utilise l'objet NumberFormatter (mx.data.formatters.NumberFormatter) inclus dans Flash MX Professionnel 2004. Cet objet de mise en forme contient une propriété appelée `precision`, qui indique le nombre de chiffres à afficher après la virgule. Ce code définit la propriété `precision` à deux chiffres après la virgule pour un champ utilisant l'objet NumberFormatter.

```
var monChamp:DataType = dataGrid.getField("soldeActuel");
monChamp.formatter.precision = 2;
```

DataType.getAsBoolean()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`objetDataType``.getAsBoolean()`

Renvoie

Une valeur booléenne.

Description

Méthode : extrait la valeur active du champ sous forme de valeur booléenne. La valeur est convertie au format booléen si nécessaire.

Exemple

Dans cet exemple, un champ appelé `NomProp` appartenant au composant `monComposant` est extrait sous forme de valeur booléenne, puis affecté à une variable.

```
var objDataType:mx.data.binding.DataType = monComposant.getField("nomProp");
var valeurProp:Boolean = objetDatatype.getAsBoolean();
```

DataType.getAsNumber()

Disponibilité

Flash Player 6.

Edition

Flash MX Professionnel 2004.

Utilisation

objetDataType.getAsNumber()

Renvoie

Un nombre.

Description

Méthode : extrait la valeur active du champ sous forme de nombre. La valeur est convertie au format numérique si nécessaire.

Exemple

Dans cet exemple, un champ appelé `NomProp` appartenant au composant `monComposant` est extrait sous forme de nombre, puis affecté à une variable.

```
var objDataType:mx.data.binding.DataType = monComposant.getField("nomProp");
var valeurProp:Number = objDataType.getAsNumber();
```

Voir aussi

DataType.getAnyTypedValue()

DataType.getAsString()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

objetDataType.getAsString()

Renvoie

Une chaîne.

Description

Méthode : extrait la valeur active du champ sous forme de chaîne. La valeur est convertie au format chaîne si nécessaire.

Exemple

Dans cet exemple, une propriété d'un composant appelé `NomProp` appartenant au composant `monComposant` est extraite sous forme de chaîne, puis affectée à une variable.

```
var objDataType:mx.data.binding.DataType = monComposant.getField("nomProp");
var valeurProp:String = objDataType.getAsString();
```

Voir aussi

`DataType.getAnyTypedValue()`

DataType.getAnyTypedValue()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`objetDataType``.getAnyTypedValue(`*`TypesSuggérés`*`)`

Paramètres

`TypesSuggérés` Tableau de chaînes spécifiant, dans l'ordre décroissant, les types de donnés préférés que vous souhaitez utiliser pour le champ. Pour plus d'informations, consultez la section Description ci-après.

Renvoie

La valeur active du champ, sous la forme d'un des types de données spécifiés dans le tableau *`TypesSuggérés`*.

Description

Méthode : extrait la valeur active du champ et la traite à l'aide des informations de schéma du champ. Si le champ est en mesure de fournir une valeur comme premier type de données spécifié dans le tableau *`TypesSuggérés`*, la méthode renvoie la valeur dans ce type de données. Dans le cas contraire, la méthode tente d'extraire la valeur du champ en utilisant le deuxième type de données spécifié dans le tableau `TypesSuggérés`, et ainsi de suite.

Si vous spécifiez `null` comme l'un des éléments dans le tableau *`TypesSuggérés`*, la méthode renvoie la valeur du champ dans le type de données spécifié dans le panneau Schéma. La spécification de `null` entraîne systématiquement le renvoi d'une valeur ; aussi, utilisez `null` uniquement à la fin du tableau.

Si une valeur ne peut pas être renvoyée sous la forme de l'un des types suggérés, elle est renvoyée sous la forme du type spécifié dans le panneau Schéma.

Exemple

Cet exemple tente d'obtenir la valeur d'un champ (`infosProduit.disponibles`) dans la propriété `results` d'un composant XMLConnector, tout d'abord sous forme de nombre, ou en cas d'échec, sous forme de chaîne.

```
import mx.data.binding.DataType;
import mx.data.binding.TypedValue;
var f: DataType = monConnecteurXml.getField("résultats",
  "infosProduit.disponibles");
var b: TypedValue = f.getAnyTypedValue(["Number", "String"]);
```

Voir aussi

`ComponentMixins.getField()`

DataType.getTypedValue()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`objetDataType``.getTypedValue(`*`TypeDemandé`*`)`

Paramètres

`TypeDemandé` Chaîne contenant le nom d'un type de données, ou `null`.

Renvoie

Un objet TypedValue (consultez *Classe TypedValue (Flash Professionnel uniquement)*, page 160)

Description

Méthode : renvoie la valeur du champ sous la forme spécifiée par *`TypeDemandé`*, s'il est spécifié et si le champ peut fournir sa valeur sous cette forme. Si le champ n'est pas en mesure de fournir sa valeur sous la forme demandée, la méthode renvoie la valeur `null`.

Si null le paramètre *`TypeDemandé`* a la valeur `null`, la méthode renvoie la valeur du champ dans son type par défaut.

Exemple

```
var bool:TypedValue = champ.getTypedValue("Booléen");
```

DataType.kind

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

objetDataType.kind

Description

Propriété : fournit une référence à l'objet Kind associé à ce champ. Vous pouvez l'utiliser pour accéder aux propriétés et aux méthodes de l'objet Kind.

DataType.setAnyTypedValue()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

objetDataType.setAnyTypedValue(*nouvelleValeur*)

Paramètres

nouvelleValeur Valeur d'un objet TypedValue à définir dans le champ.

Pour plus d'informations sur les objets TypedValue, consultez *Classe TypedValue (Flash Professionnel uniquement)*, page 160.

Renvoie

Un tableau de chaînes décrivant les erreurs qui se sont produites lors de la définition de la nouvelle valeur. Des erreurs peuvent se produire dans les cas de figure suivants :

- Les données fournies ne peuvent pas être converties dans le type de données de ce champ (par exemple, la chaîne "abc" ne peut pas être convertie au format Number).
- Les données sont d'un type acceptable mais ne répondent pas aux critères de validation du champ.
- Le champ est en lecture seule.

Remarque : Le texte des messages variera en fonction du type de données, des mises en forme et des encodeurs définis dans le schéma du champ.

Description

Méthode : définit une nouvelle valeur dans le champ, à l'aide des informations de schéma du champ, pour traiter le champ.

Cette méthode fonctionne en appelant d'abord la méthode `DataType.setTypedValue()` pour définir la valeur. Si cette opération échoue, la méthode vérifie si l'objet de destination accepte des données au format String, Boolean ou Number. Si c'est le cas, elle tente ensuite d'utiliser les fonctions de conversion ActionScript correspondantes.

Exemple

Cet exemple crée un nouvel objet TypedValue (booléen) puis affecte cette valeur à un objet DataType appelé `champ`. Les erreurs qui se produisent sont affectées au tableau `erreurs`.

```
import mx.data.binding.*;
var t:TypedValue = new TypedValue (true, "Boolean");
var erreurs: Array = champ.setAnyTypedValue (t);
```

Voir aussi

`DataType.setTypedValue()`

DataType.setAsBoolean()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`objetDataType``.setAsBoolean(`*`nouvelleValeurBoolénne`*`)`

Paramètres

`nouvelleValeurBooléenne` Valeur booléenne.

Renvoie

Rien.

Description

Méthode : définit le champ à la nouvelle valeur, qui est donnée sous forme booléenne (Boolean). La valeur est convertie au type de données approprié pour ce champ, puis stockée sous ce type de données.

Exemple

```
var bool: Boolean = true;
champ.setAsBoolean (bool);
```

DataType.setAsNumber()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

objetDataType.setAsNumber(*nouvelleValeurNumérique*)

Paramètres

nouvelleValeurNumérique Nombre.

Renvoie

Rien.

Description

Méthode : définit le champ à la nouvelle valeur, donnée sous forme numérique (Number). La valeur est convertie au type de données approprié pour ce champ, puis stockée sous ce type de données.

Exemple

```
var num: Number = 32;
champ.setAsNumber (num);
```

DataType.setAsString()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

objetDataType.setAsString(*nouvelleValeurChaîne*)

Paramètres

nouvelleValeurChaîne Chaîne.

Renvoie

Rien.

Description

Méthode : définit le champ à la nouvelle valeur, donnée sous forme de chaîne (String). La valeur est convertie au type de données approprié pour ce champ, puis stockée sous ce type de données.

Exemple

```
var stringVal: String = "La nouvelle valeur";
champ.setAsString (valeurChaîne);
```

DataType.setTypedValue()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

objetDataType.setTypedValue(*newValue*)

Paramètres

nouvelleValeur Valeur d'un objet TypedValue à définir dans le champ.

Pour plus d'informations sur les objets TypedValue, consultez *Classe TypedValue (Flash Professionnel uniquement)*, page 160.

Renvoie

Un tableau de chaînes décrivant les erreurs qui se sont produites lors de la définition de la nouvelle valeur. Des erreurs peuvent se produire dans les cas de figure suivants :

- Les données fournies ne sont pas d'un type acceptable.
- Les données fournies ne peuvent pas être converties dans le type de données de ce champ (par exemple, la chaîne "abc" ne peut pas être convertie au format Number).
- Les données sont d'un type acceptable mais ne répondent pas aux critères de validation du champ.
- Le champ est en lecture seule.

Remarque : Le texte des messages variera en fonction du type de données, des mises en forme et des encodeurs définis dans le schéma du champ.

Description

Méthode : définit une nouvelle valeur dans le champ, à l'aide des informations de schéma du champ, pour traiter le champ. Cette méthode agit de façon similaire à la méthode `DataType.setAnyTypedValue()`, sauf qu'elle ne tente pas de convertir les données en un type de données acceptable. Pour plus d'informations, consultez `DataType.setAnyTypedValue()`.

Exemple

Cet exemple crée un nouvel objet TypedValue (booléen) puis affecte cette valeur à un objet DataType appelé `champ`. Les erreurs qui se produisent sont affectées au tableau `erreurs`.

```
import mx.data.binding.*;
var bool:donnéeEntrée = new TypedValue (true, "Boolean");
var erreurs: Array = champ.setTypedValue (bool);
```

Voir aussi

`DataType.setTypedValue()`

Classe TypedValue (Flash Professionnel uniquement)

Nom de classe ActionScript mx.data.binding.TypedValue

Un objet TypedValue contient une valeur de données, ainsi que des informations sur le type de données de cette valeur. Les objets TypedValue sont utilisés en tant que paramètres et renvoyés par plusieurs méthodes de la classe DataType. Les informations de type de données de l'objet TypedValue permettent aux objets DataType de déterminer quand et comment ils doivent effectuer la conversion du type de données.

Remarque : Pour rendre cette classe disponible lors de l'exécution, vous devez inclure le composant DataBindingClasses dans votre document FLA. Pour plus d'informations, consultez « Utilisation de la liaison des données et des services web à l'exécution (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Pour un aperçu des classes dans le paquet mx.data.binding, consultez *Classes de liaison des données (Flash Professionnel uniquement)*, page 128.

Propriétés de la classe TypedValue

Propriété	Description
`TypedValue.type`	Contient le schéma associé à la valeur de l'objet TypedValue.
`TypedValue.typeName`	Contient le paramètre DataType de la valeur de l'objet TypedValue.
`TypedValue.value`	Contient la valeur de données de l'objet TypedValue.

Constructeur de la classe TypedValue

Disponibilité

Flash Player 6 version 79.

Usage

```
new mx.data.binding.TypedValue(valeur, nomType, [type])
```

Paramètres

`valeur` Valeur de données pouvant être de tout type.

`nomType` Chaîne contenant le paramètre DataType de la valeur.

`type` (Facultatif) Objet Schema décrivant de façon détaillée le schéma des données. Ce champ est requis uniquement dans certaines circonstances ; par exemple, lors de la définition des données dans la propriété `dataProvider` d'un composant DataSet.

Description

Constructeur : crée un nouvel objet TypedValue.

TypedValue.type

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

objetTypedValue.type

Description

Propriété : contient le schéma associé à la valeur de l'objet TypedValue. Elle est utilisée uniquement dans certaines circonstances.

Exemple

Cet exemple affichera "null" dans le panneau de sortie.

```
var t: valeurEntrée = new TypedValue (true, "Boolean", null);
trace(t.type);
```

TypedValue.typeName

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

objetTypedValue.typeName

Description

Propriété : contient le paramètre DataType de la valeur de l'objet TypedValue.

Exemple

Cet exemple affichera "Boolean" dans le panneau de sortie.

```
var t: valeurEntrée = new TypedValue (true, "Boolean", null);
trace(t.typeName);
```

TypedValue.value

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

objetTypedValue.value

Description

Propriété : contient la valeur de données de l'objet TypedValue.

Exemple

Cet exemple affichera "true" dans le panneau de sortie.

```
var t: valeurEntrée = new TypedValue (true, "Boolean", null);
trace(t.value);
```

Composant DataGrid (Flash Professionnel uniquement)

Le composant DataGrid vous permet de créer des applications et des affichages de données puissants. Vous pouvez utiliser le composant DataGrid pour instancier un jeu d'enregistrements (extrait d'une requête de base de données dans ColdFusion, Java ou .Net) à l'aide de Macromedia Flash Remoting et l'afficher en colonnes. Vous pouvez également utiliser les données d'un ensemble de données ou d'un tableau pour remplir un composant DataGrid. Le composant v2 DataGrid a été amélioré afin d'inclure la fonction de défilement horizontal, une meilleure prise en charge des événements (notamment des événements dans les cellules modifiables), des fonctionnalités de tri plus poussées, ainsi qu'une optimisation des performances.

Vous pouvez redimensionner et personnaliser des caractéristiques telles que la police, la couleur et les bordures des colonnes d'une grille. Vous pouvez utiliser un clip personnalisé en tant qu'objet Cell Renderer (pour le rendu des cellules) pour les colonnes d'une grille. Un objet Cell Renderer affiche le contenu d'une cellule. Vous pouvez utiliser les barres de défilement pour faire défiler les données figurant dans une grille ; vous pouvez également désactiver les barres de défilement et utiliser les méthodes DataGrid pour créer un affichage du style mode Page.

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

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

Vous activez l'accessibilité d'un composant en une seule fois, quel que soit le nombre de ses occurrences. Pour plus d'informations, consultez « Création de contenu accessible », dans le guide Utilisation de Flash de l'aide.

Interaction avec le composant DataGrid (Flash Professionnel uniquement)

Vous pouvez utiliser la souris et le clavier pour interagir avec un composant DataGrid.

Si `DataGrid.sortableColumns` est `true` et `DataGridColumn.sortOnHeaderRelease` est `true`, l'utilisateur peut trier la grille en fonction des valeurs des cellules d'une colonne en cliquant sur l'en-tête de cette dernière.

Si `DataGrid.resizableColumns` est `true`, l'utilisateur peut redimensionner les colonnes en cliquant sur la zone entre les colonnes.

Si l'utilisateur clique sur une cellule modifiable, cette cellule a le focus ; s'il clique sur une cellule non modifiable, cela n'a aucun impact sur le focus. Une cellule est modifiable lorsque ses propriétés `DataGrid.editable` et `DataGridColumn.editable` ont la valeur `true`.

Lorsqu'une occurrence de DataGrid a le focus (après que l'utilisateur a cliqué ou utilisé la touche de tabulation), il est possible d'utiliser les touches suivantes pour la contrôler :

Touche	Description
Flèche vers le bas	Lorsque la cellule fait l'objet d'une modification, le point d'insertion se positionne à la fin du texte de la cellule. Si la cellule n'est pas modifiable, la flèche vers le bas gère la sélection de la même façon que le composant List.
Flèche vers le haut	Lorsque la cellule fait l'objet d'une modification, le point d'insertion se positionne au début du texte de la cellule. Si la cellule n'est pas modifiable, la flèche vers le haut gère la sélection de la même façon que le composant List.
Flèche vers la droite	Lorsque la cellule fait l'objet d'une modification, le point d'insertion se déplace d'un caractère vers la droite. Si la cellule n'est pas modifiable, cette action n'a aucune incidence.
Flèche vers la gauche	Lorsque la cellule fait l'objet d'une modification, le point d'insertion se déplace d'un caractère vers la gauche. Si la cellule n'est pas modifiable, cette action n'a aucune incidence.
Retour/Entrée/ Maj+Entrée	Lorsque la cellule est modifiable, la modification est validée et le point d'insertion se place dans la cellule de la ligne suivante (vers le haut ou vers le bas, en fonction du sens de basculement) dans la même colonne.
Maj+Tab/Tab	Place le focus sur l'élément précédent. Lorsque l'utilisateur appuie sur la touche Tab, le focus englobe de la dernière colonne de la grille jusqu'à la première colonne de la ligne suivante. Lorsque l'utilisateur appuie sur Maj+Tab, l'ordre en inversé.

Utilisation du composant DataGrid (Flash Professionnel uniquement)

Vous pouvez utiliser le composant DataGrid comme base pour de nombreux types d'applications de données. Vous pouvez facilement créer une vue tabulaire d'une requête de base de données (ou d'autres données) et utiliser les fonctionnalités du composant CellRenderer pour créer des éléments d'interface utilisateur plus sophistiqués et modifiables. Voici des exemples pratiques d'utilisation du composant DataGrid :

- Client de messagerie web
- Pages de résultats de recherches
- Tableurs (calculateurs d'emprunts et applications de formulaires de déclaration d'impôt)

Le composant DataGrid comprend deux ensembles d'API : la classe DataGrid et la classe DataGridColumn.

Présentation du composant DataGrid : affichage et modèle de données

Basiquement, le composant DataGrid est constitué d'un modèle de données et d'un affichage de présentation des données. Le modèle de données est constitué de trois composants principaux :

- DataProvider (Fournisseur de données)

Il s'agit d'une liste d'éléments permettant de remplir la grille de données. Un tableau se trouvant dans la même image qu'un composant DataGrid se voit automatiquement attribuer des méthodes (de l'API DataProvider) qui permettent de manipuler des données et de diffuser les modifications dans plusieurs affichages. Tout objet qui implémente l'interface DataProvider peut être affecté à la propriété `DataGrid.dataProvider` (notamment des jeux d'enregistrements ou des ensembles de données). Le code suivant crée un fournisseur de données appelé `monDP` :

```
monDP = new Array({nom:"Chris", prix:"SansPrix"}, {nom:"Nigel",
  prix:"Economique"});
```

- Elément

  Il s'agit d'un objet ActionScript utilisé pour stocker les unités d'informations des cellules d'une colonne. Une grille de données est en fait une liste pouvant afficher plusieurs colonnes de données. Une liste fonctionne de façon similaire à un tableau ; chaque espace indexé de la liste est appelé élément. Pour le composant DataGrid, chaque élément est composé de champs. Dans le code suivant, le texte placé entre accolades (`{}`) est un élément :

  ```
  monDP = new Array({nom:"Chris", prix:"SansPrix"}, {nom:"Nigel",
    prix:"Economique"});
  ```

- Champ

  Identificateur indiquant le nom des colonnes dans les éléments. Correspond à la propriété `columnNames` dans la liste de colonnes. Habituellement le composant List utilise les champs `label` et `data` ; dans le cas du composant DataGrid, il peut s'agir de n'importe quel identificateur. Dans le code suivant, les champs sont `nom` et `prix` :

  ```
  monDP = new Array({nom:"Chris", prix:"SansPrix"}, {nom:"Nigel",
    prix:"Economique"});
  ```

La vue comporte trois parties principales :

- Ligne (row)

  Il s'agit d'un objet affichage chargé du rendu des éléments de la grille grâce à la disposition des cellules. Chaque ligne est disposée horizontalement sous la ligne précédente.

- Colonne (Column)

  Il s'agit des objets affichage (occurrences de la classe DataGridColumn) chargés d'afficher chaque colonne (elles contiennent les données de largeur, couleur, taille etc.).

  Il existe trois méthodes pour ajouter des colonnes à une grille de données : L'affectation d'un objet DataProvider à `DataGrid.dataProvider` (cette opération génère automatiquement une colonne pour chaque champ dans le premier élément), la définition de la propriété `DataGrid.columnNames`, pour préciser les champs qui seront affichés, ou l'utilisation du constructeur de la classe DataGridColumn pour créer des colonnes, suivi de l'appel de la méthode `DataGrid.addColumn()` pour les ajouter à la grille.

  Pour formater les colonnes, vous pouvez définir les propriétés de style de toute la grille de données ou définir les objets DataGridColumn, configurer leur format de style de façon individuelle puis les ajouter à la grille de données.

- Cellule (Cell)

  Il s'agit d'un objet affichage chargé du rendu de chaque champ de chaque élément. Pour communiquer avec la grille de données, ces composants doivent implémenter l'interface CellRenderer (consultez *API CellRenderer*, page 84). Dans une grille de données de base, une cellule est un objet TextField ActionScript intégré.

Paramètres du composant DataGrid

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

multipleSelection Valeur booléenne indiquant si plusieurs éléments peuvent être sélectionnés (`true`) ou non (`false`). La valeur par défaut est `false`.

rowHeight Hauteur de chaque ligne, en pixels. La modification de la taille de la police ne modifie pas la hauteur de la ligne. La valeur par défaut est 20.

editable Valeur booléenne indiquant si la grille est modifiable (`true`) ou non (`false`). La valeur par défaut est `false`.

Vous pouvez contrôler ces options et d'autres options du composant DataGrid à l'aide des propriétés, méthodes et événements d'ActionScript. Pour plus d'informations, consultez *Classe DataGrid (Flash Professionnel uniquement)*, page 167.

Création d'une application avec le composant DataGrid

Pour créer une application avec le composant DataGrid, vous devez tout d'abord déterminer d'où viennent vos données. Les données d'une grille peuvent provenir d'un jeu d'enregistrements alimenté par une requête de base de données dans Macromedia ColdFusion, Java, ou .Net à l'aide du système Flash Remoting. Les données peuvent également provenir d'un ensemble de données ou d'un tableau. Pour les intégrer dans une grille, vous devez définir la propriété `DataGrid.dataProvider` (jeu d'enregistrements, ensemble de données ou tableau correspondant). Vous pouvez également utiliser les méthodes des classes DataGrid et DataGridColumn pour créer des données localement. Un objet Array (tableau) se trouvant dans la même image qu'un composant DataGrid copie les méthodes, propriétés et événements de la classe DataProvider.

Pour utiliser le système Flash Remoting afin d'ajouter un composant DataGrid à une application, procédez comme suit :

1. Dans Flash, choisissez Fichier > Nouveau et sélectionnez Document Flash.
2. Dans le panneau des composants, double-cliquez sur le composant DataGrid pour l'ajouter sur la scène.
3. Dans l'inspecteur des propriétés, entrez **maGrilleDeDonnées** comme nom d'occurrence.
4. Dans le panneau Actions, sur l'Image 1, entrez le code suivant :

   ```
   maGrilleDeDonnées.dataProvider = occurrenceJeuEnregistrements;
   ```

   Le jeu d'enregistrements Flash Remoting `occurrenceJeuEnregistrements` est affecté à la propriété `dataProvider` de `maGrilleDeDonnées`.
5. Choisissez Contrôle > Tester l'animation.

Pour utiliser un fournisseur de données local afin d'ajouter un composant DataGrid à une application :

1. Dans Flash, choisissez Fichier > Nouveau et sélectionnez Document Flash.
2. Dans le panneau des composants, double-cliquez sur le composant DataGrid pour l'ajouter sur la scène.
3. Dans l'inspecteur des propriétés, entrez **maGrilleDeDonnées** comme nom d'occurrence.

4 Dans le panneau Actions, sur l'Image 1, entrez le code suivant :

```
monDP = new Array({nom:"Chris", prix:"SansPrix"}, {nom:"Nigel",
  prix:"Economique"});
maGrilleDeDonnées.dataProvider = monDP;
```

Les champs `nom` et `prix` sont utilisés comme en-têtes de colonne et leurs valeurs remplissent les cellules dans chaque ligne.

5 Choisissez Contrôle > Tester l'animation.

Personnalisation du composant DataGrid (Flash Professionnel uniquement)

Vous pouvez transformer un composant DataGrid horizontalement et verticalement durant la programmation et à l'exécution. Lors de la programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez la méthode `setSize()` (consultez `UIObject.setSize()`). Si aucune barre de défilement horizontale n'est présente, la largeur des colonnes s'ajuste proportionnellement. Si une modification de la taille des colonnes (et donc des cellules) intervient, le texte des cellules peut être tronqué.

Utilisation des styles avec le composant DataGrid

Vous pouvez définir des propriétés de style afin de modifier l'aspect d'un composant DataGrid. Le composant DataGrid hérite des styles de halo du composant List. Pour plus d'informations, consultez *Utilisation des styles avec le composant List*, page 308. Le composant DataGrid prend également en charge les styles de halo suivants :

Style	Description
`backgroundColor`	La couleur d'arrière-plan peut être définie pour l'ensemble de la grille ou pour chaque colonne.
`labelStyle`	Le style de police peut être défini pour l'ensemble de la grille ou pour chaque colonne.
`headerStyle`	Une déclaration de style CSS pour l'en-tête de colonne, pouvant être appliquée à une grille ou à une colonne.
`vGridLines`	Valeur booléenne indiquant si les lignes verticales de la grille doivent être affichées (`true`) ou non (`false`).
`hGridLines`	Valeur booléenne indiquant si les lignes horizontales de la grille doivent être affichées (`true`) ou non (`false`).
`vGridLineColor`	Couleur des lignes verticales de la grille.
`hGridLineColor`	Couleur des lignes horizontales de la grille.
`headerColor`	Couleur des en-têtes de colonnes.

Si le tableau ci-dessus indique qu'un style peut être défini pour une colonne, vous pouvez utiliser la syntaxe suivante pour définir le style :

```
grid.getColumnAt(3).setStyle("backgroundColor", 0xff00aa)
```

Utilisation d'enveloppes avec le composant DataGrid

Les enveloppes que le composant DataGrid utilise pour représenter ses états virtuels sont incluses dans les sous-composants dont la grille de données est composée (ScrollPane et RectBorder). Pour plus d'informations sur leurs enveloppes, consultez *Utilisation des enveloppes avec le composant ScrollPane*, page 492 et *Utilisation des enveloppes avec le composant List*, page 309.

Toutefois, les sous-couches de sélection et de survol utilisent l'API de dessin d'ActionScript. Pour appliquer des enveloppes à ces portions de la grille de données au cours de la programmation, modifiez le code ActionScript dans les symboles d'enveloppes, dans le dossier Flash UI Components 2/Themes/MMDefault/datagrid/skins states, dans la bibliothèque de l'un des fichiers FLA de thèmes. Pour plus d'informations, consultez *A propos de l'application des enveloppes aux composants*, page 38.

Classe DataGrid (Flash Professionnel uniquement)

Héritage mx.core.UIObject > mx.core.UIComponent > mx.core.View > mx.core.ScrollView > mx.controls.listclasses.ScrollSelectList > mx.controls.List

Nom de classe ActionScript mx.controls.DataGrid

Toutes les classes de composants ont une propriété `version` qui correspond à une propriété de classe. Les propriétés de classe 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.DataGrid.version);
```

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

Méthodes de la classe DataGrid

Méthode	Description
`DataGrid.addColumn()`	Ajoute une colonne à la grille de données.
`DataGrid.addColumnAt()`	Ajoute une colonne à la grille de données à un emplacement spécifique.
`DataGrid.addItem()`	Ajoute un élément à la grille de données.
`DataGrid.addItemAt()`	Ajoute un élément à la grille de données à un emplacement spécifique.
`DataGrid.editField()`	Remplace les données d'une cellule à un emplacement spécifié.
`DataGrid.getColumnAt()`	Obtient une référence à une colonne à un emplacement spécifié.
`DataGrid.getColumnIndex()`	Obtient l'index de la colonne.
`DataGrid.removeAllColumns()`	Supprime toutes les colonnes d'une grille de données.
`DataGrid.removeColumnAt()`	Supprime une colonne d'une grille de données à un emplacement spécifié.
`DataGrid.replaceItemAt()`	Remplace un élément à un emplacement spécifié par un autre élément.
`DataGrid.spaceColumnsEqually()`	Espace les colonnes de manière égale.

Hérite de toutes les propriétés des classes *UIObject* et *UIComponent*.

Propriétés de la classe DataGrid

Propriété	Description
`DataGrid.columnCount`	Lecture seule. Nombre de colonnes affichées.
`DataGrid.columnNames`	Tableau des noms de champs (affichés sous forme de colonnes) de chaque élément.
`DataGrid.dataProvider`	Modèle de données d'une grille de données.
`DataGrid.editable`	Valeur booléenne indiquant si la grille de données est modifiable (`true`) ou non (`false`).
`DataGrid.focusedCell`	Définit la cellule qui a le focus.
`DataGrid.headerHeight`	Hauteur des en-têtes de colonnes, en pixels.
`DataGrid.hScrollPolicy`	Indique si une barre de défilement horizontale est présente (`"on"`), non présente (`"off"`) ou apparaît lorsque cela est nécessaire (`"auto"`).
`DataGrid.resizableColumns`	Valeur booléenne indiquant si les colonnes peuvent être redimensionnées (`true`) ou non (`false`).
`DataGrid.selectable`	Valeur booléenne indiquant si la grille de données peut être sélectionnée (`true`) ou non (`false`).
`DataGrid.showHeaders`	Valeur booléenne indiquant si les en-têtes de colonnes sont visibles (`true`) ou non (`false`).
`DataGrid.sortableColumns`	Valeur booléenne indiquant si les colonnes peuvent être triées (`true`) ou non (`false`).

Evénements de la classe DataGrid

Evénement	Description
`DataGrid.cellEdit`	Diffusé lorsque la valeur de la cellule a changé.
`DataGrid.cellFocusIn`	Diffusé lorsqu'une cellule reçoit le focus.
`DataGrid.cellFocusOut`	Diffusé lorsqu'une cellule perd le focus.
`DataGrid.cellPress`	Diffusé lorsque l'utilisateur clique dans une cellule.
`DataGrid.change`	Diffusé lorsqu'un élément a été sélectionné.
`DataGrid.columnStretch`	Diffusé lorsqu'une colonne est redimensionnée par un utilisateur.
`DataGrid.headerRelease`	Diffusé lorsqu'un utilisateur clique sur un en-tête puis le désélectionne.

DataGrid.addColumn()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDeDonnées.addColumn(colonneGrilleDeDonnées)
maGrilleDeDonnées.addColumn(nom)
```

Paramètres

colonneGrilleDeDonnées Occurrence de la classe DataGridColumn.

nom Chaîne indiquant le nom d'un nouvel objet DataGridColumn à insérer.

Renvoie

Une référence à l'objet DataGridColumn qui a été ajouté.

Description

Méthode : ajoute une nouvelle colonne à la fin de la grille de données. Pour plus d'informations, consultez *Classe DataGridColumn (Flash Professionnel uniquement)*, page 187.

Exemple

Le code suivant permet d'ajouter un nouvel objet DataGridColumn nommé `Violet` :

```
import mx.controls.gridclasses.DataGridColumn;
maGrille.addColumn(new DataGridColumn("Violet"));
```

DataGrid.addColumnAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

Usage 1 :

```
maGrilleDeDonnées.addColumnAt(index, nom)
```

Usage 2 :

```
maGrilleDeDonnées.addColumnAt(index, colonneGrilleDeDonnées)
```

Paramètres

index Position d'index à laquelle l'objet DataGridColumn est ajouté. La première position est 0.

nom Chaîne indiquant le nom de l'objet DataGridColumn. Vous devez spécifier le paramètre *index* ou le paramètre *colonneGrilleDeDonnées*.

colonneGrilleDeDonnées Occurrence de la classe DataGridColumn.

Renvoie

Une référence à l'objet DataGridColumn qui a été ajouté.

Description

Méthode : ajoute une nouvelle colonne à l'emplacement spécifié. Les colonnes sont déplacées vers la droite et leur index est incrémenté. Pour plus d'informations, consultez *Classe DataGridColumn (Flash Professionnel uniquement)*, page 187.

Exemple

Dans l'exemple suivant, un nouvel objet DataGridColumn nommé `"Vert"` est inséré au niveau de la deuxième et de la quatrième colonne :

```
import mx.controls.gridclasses.DataGridColumn;
maGrille.addColumnAt(1, "Vert");
maGrille.addColumnAt(3, new DataGridColumn("Violet"));
```

DataGrid.addItem()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDeDonnées.addItem(élément)
```

Paramètres

élément Occurrence d'un objet à ajouter à la grille.

Renvoie

Une référence à l'occurrence qui a été ajoutée.

Description

Méthode : ajoute un élément à la fin de la grille (après le dernier index d'élément).

Remarque : Cette méthode diffère de la méthode `List.addItem()` dans la mesure où c'est un objet, et non une chaîne, qui est transmis.

Exemple

L'exemple suivant permet d'ajouter un nouvel objet dans la grille `maGrille` :

```
var objet= {name:"Jim!!", age:30};
var objetAjouté = maGrille.addItem(objet);
```

DataGrid.addItemAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDeDonnées.addItemAt(index, élément)
```

Paramètres

index Ordre (parmi les nœuds enfants) dans lequel le nœud doit être ajouté. La première position est 0.

élément Chaîne qui affiche le nœud.

Renvoie

Une référence à l'occurrence d'objet qui a été ajoutée.

Description

Méthode : ajoute un élément à la grille à l'emplacement spécifié.

Exemple

L'exemple suivant permet d'insérer une occurrence d'objet dans la grille à la position d'index 4 :

```
var objet= {name:"Jim!!", age:30};
var objetAjouté = maGrille.addItemAt(4, objet);
```

DataGrid.cellEdit

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
objetDécoute.cellEdit = function(objetEvt){
  // insérez votre code ici
}
occurrenceMaGrilleDeDonnées.addEventListener("cellEdit", objetDécoute)
```

Description

Evénement : diffusé à tous les objets d'écoute enregistrés lorsque la valeur d'une cellule a changé.

Les composants V2 utilisent un modèle d'événement dispatcher/écouteur. Le composant DataGrid distribue un événement `cellEdit` lorsque la valeur d'une cellule a été modifiée ; l'événement est géré par une fonction (également appelée *gestionnaire*) associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`objetEvt`*) au gestionnaire. 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. L'objet événement de l'événement `DataGrid.cellEdit` possède quatre propriétés supplémentaires :

`columnIndex` Nombre indiquant l'index de la colonne cible.

`itemIndex` Nombre indiquant l'index de la ligne cible.

`oldValue` Valeur précédente de la cellule.

`type` La chaîne `"cellEdit"`.

Pour plus d'informations, consultez *Objets événement*, page 592.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `EcouteurDeMaGrilleDeDonnées` est défini puis transmis à la méthode `maGrilleDeDonnées.addEventListener()` en tant que second paramètre. L'objet événement est capturé par le gestionnaire `cellEdit` dans le paramètre *`objetEvt`*. Lorsque l'événement `cellEdit` est diffusé, une instruction `trace` est envoyée vers le panneau de sortie, comme suit :

```
EcouteurDeMaGrilleDeDonnées = new Object();
EcouteurDemaGrilleDeDonnées.cellEdit = function(event){
  var cellule = "(" + event.columnIndex + ", " + event.itemIndex + ")";
  trace("La valeur de la cellule au niveau de " + cell + "a changé");
}
maGrilleDeDonnées.addEventListener("cellEdit", EcouteurDeMaGrilleDeDonnées);
```

Remarque : La grille doit être modifiable pour que ce code puisse fonctionner.

DataGrid.cellFocusIn

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
objetDécoute.cellFocusIn = function(objetEvt){
  // insérez votre code ici
}
occurrenceMaGrilleDeDonnées.addEventListener("cellFocusIn", objetDécoute)
```

Description

Evénement : diffusé à tous les objets d'écoute enregistrés lorsqu'une cellule reçoit le focus. Cet événement est diffusé une fois que les événements `editCell` et `cellFocusOut` d'une cellule préalablement modifiée sont diffusés.

Les composants V2 utilisent un modèle d'événement dispatcher/écouteur. Lorsqu'un composant DataGrid distribue un événement `cellFocusIn`, cet événement est géré par une fonction (également appelée *gestionnaire*) associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`objetEvt`*) au gestionnaire. 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. L'objet événement de l'événement `DataGrid.cellFocusIn` possède trois propriétés supplémentaires :

`columnIndex` Nombre indiquant l'index de la colonne cible.

`itemIndex` Nombre indiquant l'index de la ligne cible.

`type` La chaîne `"cellFocusIn"`.

Pour plus d'informations, consultez *Objets événement*, page 592.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `monEcouteur` est défini puis transmis à la méthode `grid.addEventListener()` comme second paramètre. L'objet événement est capturé par le gestionnaire `cellFocusIn` dans le paramètre *`objetEvt`*. Lorsque l'événement `cellFocusIn` est diffusé, une instruction `trace` est envoyée vers le panneau de sortie, comme suit :

```
var monEcouteur = new Object();
monEcouteur.cellFocusIn = function(event) {
    var cellule = "(" + event.columnIndex + ", " + event.itemIndex + ")";
    trace("La cellule au niveau de " + cell + " a obtenu le focus");
};
grid.addEventListener("cellFocusIn", monEcouteur);
```

Remarque : La grille doit être modifiable pour que ce code puisse fonctionner.

DataGrid.cellFocusOut

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
objetDécoute.cellFocusOut = function(objetEvt){
  // insérez votre code ici
}
occurrenceMaGrilleDeDonnées.addEventListener("cellFocusOut", objetDécoute)
```

Description

Evénement : diffusé à tous les objets d'écoute enregistrés chaque fois qu'un utilisateur quitte une cellule qui a le focus. Vous pouvez utiliser les propriétés de l'objet événement pour isoler la cellule qui a été quittée. Cet événement est diffusé une fois que l'événement `cellEdit` est diffusé et avant que les événements `cellFocusIn` consécutifs ne soient diffusés par la cellule suivante.

Les composants V2 utilisent un modèle d'événement dispatcher/écouteur. Lorsqu'un composant DataGrid distribue un événement `cellFocusOut`, celui-ci est géré par une fonction (également appelée *gestionnaire*) associée à un objet d'écoute que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`objetEvt`*) au gestionnaire. 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. L'objet événement de l'événement `DataGrid.cellFocusOut` possède trois propriétés supplémentaires :

`columnIndex` Nombre indiquant l'index de la colonne cible. La première position est 0.

`itemIndex` Nombre indiquant l'index de la ligne cible. La première position est 0.

`type` La chaîne `"cellFocusOut"`.

Pour plus d'informations, consultez *Objets événement*, page 592.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `monEcouteur` est défini puis transmis à la méthode `grid.addEventListener()` comme second paramètre. L'objet événement est capturé par le gestionnaire `cellFocusOut` dans le paramètre *`objetEvt`*. Lorsque l'événement `cellFocusOut` est diffusé, une instruction `trace` est envoyée vers le panneau de sortie, comme suit :

```
var monEcouteur = new Object();
monEcouteur.cellFocusOut = function(event) {
    var cellule = "(" + event.columnIndex + ", " + event.itemIndex + ")";
    trace("La cellule au niveau de" + cell + " a le focus");
};
grille.addEventListener("cellFocusOut", monEcouteur);
```

Remarque : La grille doit être modifiable pour que ce code puisse fonctionner.

DataGrid.cellPress

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
objetDécoute.cellPress = function(objetEvt){
  // insérez votre code ici
}
occurrenceMaGrilleDeDonnées.addEventListener("cellPress", objetDécoute)
```

Description

Evénement : diffusé à tous les objets d'écoute enregistrés lorsqu'un utilisateur clique avec la souris dans une cellule.

Les composants V2 utilisent un modèle d'événement dispatcher/écouteur. Lorsqu'un composant DataGrid diffuse un événement `cellPress`, cet événement est géré par une fonction (également appelée *gestionnaire*) associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`objetEvt`*) au gestionnaire. 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. L'objet événement de l'événement `DataGrid.cellPress` possède trois propriétés supplémentaires :

`columnIndex` Nombre indiquant l'index de la colonne cible. La première position est 0.

`itemIndex` Nombre indiquant l'index de la ligne cible. La première position est 0.

`type` La chaîne `"cellPress"`.

Pour plus d'informations, consultez *Objets événement*, page 592.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `monEcouteur` est défini puis transmis à la méthode `grid.addEventListener()` comme second paramètre. L'objet événement est capturé par le gestionnaire `cellPress` dans le paramètre *`objetEvt`*. Lorsque l'événement `cellPress` est diffusé, une instruction `trace` est envoyée vers le panneau de sortie, comme suit :

```
var monEcouteur = new Object();
monEcouteur.cellPress = function(event) {
    var cellule = "(" + event.columnIndex + ", " + event.itemIndex + ")";
    trace("L'utilisateur a cliqué sur la cellule au niveau de" + cell);
};
grid.addEventListener("cellPress", monEcouteur);
```

DataGrid.change

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
objetDécoute.change = function(objetEvt){
  // insérez votre code ici
}
occurrenceMaGrilleDeDonnées.addEventListener("change", objetDécoute)
```

Description

Evénement : diffusé à tous les objets d'écoute enregistrés lorsqu'un élément a été sélectionné.

Les composants V2 utilisent un modèle d'événement dispatcher/écouteur. Lorsqu'un composant DataGrid distribue un événement `change`, cet événement est géré par une fonction (également appelée *gestionnaire*) associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`objetEvt`*) au gestionnaire. 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. L'objet événement de l'événement `DataGrid.change` comprend une propriété supplémentaire, `type`, dont la valeur est `"change"`. Pour plus d'informations, consultez *Objets événement*, page 592.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `monEcouteur` est défini puis transmis à la méthode `grid.addEventListener()` comme second paramètre. L'objet événement est capturé par le gestionnaire `change` dans le paramètre *`objetEvt`*. Lorsque l'événement `change` est diffusé, une instruction `trace` est envoyée au panneau Sortie, comme suit :

```
var monEcouteur = new Object();
monEcouteur.change = function(event) {
    trace("La sélection a été remplacée par " + event.target.selectedIndex);
};
grille.addEventListener("change", monEcouteur);
```

DataGrid.columnCount

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDeDonnées.columnCount
```

Description

Propriété (lecture seule) : nombre de colonnes affichées.

Exemple

L'exemple suivant obtient le nombre de colonnes affichées dans l'occurrence DataGrid nommée `grille`.

```
var c = grille.columnCount;
```

DataGrid.columnNames

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDeDonnées.columnNames
```

Description

Propriété : tableau des noms de champs (affichés sous forme de colonnes) de chaque élément.

Exemple

L'exemple suivant permet d'indiquer à l'occurrence `grille` de n'afficher que ces trois champs comme colonnes :

```
grille.columnNames = ["Nom", "Description", "Prix"];
```

DataGrid.columnStretch

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
objetDécoute.columnStretch = function(objetEvt){
  // insérez votre code ici
}
occurrenceMaGrilleDeDonnées.addEventListener("columnStretch", objetDécoute)
```

Description

Evénement : diffusé à l'ensemble des écouteurs enregistrés lorsque un utilisateur redimensionne une colonne horizontalement.

Les composants V2 utilisent un modèle d'événement dispatcher/écouteur. Lorsqu'un composant DataGrid distribue un événement `columnStretch`, cet événement est géré par une fonction (également appelée *gestionnaire*) associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`objetEvt`*) au gestionnaire. 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. L'objet événement de l'événement `DataGrid.columnStretch` possède deux propriétés supplémentaires :

`columnIndex` Nombre indiquant l'index de la colonne cible. La première position est 0.

`type` La chaîne `"columnStretch"`.

Pour plus d'informations, consultez *Objets événement*, page 592.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `monEcouteur` est défini puis transmis à la méthode `grid.addEventListener()` comme second paramètre. L'objet événement est capturé par le gestionnaire `columnStretch` dans le paramètre *`objetEvt`*. Lorsque l'événement `columnStretch` est diffusé, une instruction `trace` est envoyée vers le panneau de sortie, comme suit :

```
var monEcouteur = new Object();
monEcouteur.columnStretch = function(event) {
    trace("la colonne " + event.columnIndex + "a été redimensionnée");
};
grille.addEventListener("columnStretch", monEcouteur);
```

DataGrid.dataProvider

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDeDonnées.dataProvider
```

Description

Propriété : modèle de données des éléments affichés dans un composant DataGrid.

La grille de données ajoute des méthodes au prototype de la classe Array de façon à ce que chaque objet Array soit conforme à l'interface DataProvider (voir le fichier DataProvider.as dans le dossier Classes/mx/controls/listclasses). Tout tableau existant dans la même image ou le même écran qu'une grille de données dispose automatiquement de toutes les méthodes (`addItem()`, `getItemAt()`, etc.) requises pour en faire le modèle de données d'une grille de données. Il peut être utilisé pour transmettre les modifications de modèle de données à plusieurs composants.

Dans un composant DataGrid, vous spécifiez les champs à afficher dans la propriété `DataGrid.columnNames`.

Si vous ne définissez pas l'ensemble de colonnes (en définissant la propriété `DataGrid.columnNames` ou en appelant la méthode `DataGrid.addColumn()`) de la grille de données avant que la propriété `DataGrid.dataProvider` ne soit définie, la grille de données génère des colonnes pour chaque champ dans le premier élément du fournisseur de données, une fois cet élément généré.

Tout objet implémentant l'interface DataProvider peut être utilisé en tant que fournisseur de données pour une grille de données (y compris les jeux d'enregistrements, les ensembles de données et les tableaux Flash Remoting).

Exemple

L'exemple suivant permet de créer un tableau à utiliser comme fournisseur de données et d'assigner directement ce dernier à la propriété `dataProvider` :

```
grille.dataProvider = [{nom:"Chris", prix:"Sans prix"}, {nom:"Nigel",
  Prix:"Economique"}];
```

L'exemple suivant crée un nouvel objet Array (tableau) qui est décoré avec la classe DataProvider. Il utilise une boucle `for` pour ajouter 20 éléments à la grille :

```
monFD = new Array();
for (var i=0; i<20; i++)
  monDP.addItem({nom:"Nivesh", prix:"Sans prix"});
liste.dataProvider = monDP
```

DataGrid.editable

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDeDonnées.editable
```

Description

Propriété : détermine si la grille de données peut être modifiée par un utilisateur (`true`) ou non (`false`). Cette propriété doit avoir la valeur `true` pour que les colonnes puissent être modifiées individuellement et pour que les cellules puissent recevoir le focus. La valeur par défaut est `false`.

Exemple

L'exemple suivant permet de définir la position de défilement en haut de l'affichage :

```
maGrilleDeDonnées.editable = true;
```

DataGrid.editField()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDeDonnées.editField(index, nomCol, données)
```

Paramètres

`index` Index de la cellule cible. La numérotation de cette valeur commence à zéro.

`nomCol` Chaîne indiquant le nom de la colonne (champ) contenant la cellule cible.

`données` Valeur à stocker dans la cellule cible. Ce paramètre peut être de tout type.

Renvoie

Les données qui étaient dans la cellule.

Description

Méthode : remplace les données de la cellule à l'emplacement spécifié.

Exemple

L'exemple suivant permet de placer une valeur dans la grille :

```
var valeur = maGrille.editField(5, "Nom", "Neo");
```

DataGrid.focusedCell

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDeDonnées.focusedCell
```

Description

Propriété : en mode d'édition uniquement, occurrence d'objet définissant la cellule qui a le focus. L'objet doit avoir les champs `columnIndex` et `itemIndex`, qui sont tous deux des entiers indiquant l'index de la colonne et de l'élément de la cellule. L'origine est (0,0). La valeur par défaut est undefined.

Exemple

L'exemple suivant permet de placer la cellule en surbrillance (ayant le focus) dans la quatrième ligne de la troisièème colonne :

```
grille.focusedCell = {columnIndex:2, itemIndex:3};
```

DataGrid.getColumnAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDeDonnées.getColumnAt(index)
```

Paramètres

index Index de l'objet DataGridColumn à renvoyer. La numérotation de cette valeur commence à zéro.

Renvoie

Un objet DataGridColumn.

Description

Méthode : obtient une référence à l'objet DataGridColumn au niveau de l'index spécifié.

Exemple

L'exemple suivant obtient l'objet DataGridColumn à l'index 4 :

```
var colonne = maGrille.getColumnAt(4);
```

DataGrid.getColumnIndex()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDeDonnées.getColumnIndex(index)
```

Paramètres

index Index de l'objet DataGridColumn à renvoyer.

Renvoie

Un objet DataGridColumn.

Description

Méthode : obtient une référence à l'objet DataGridColumn au niveau de l'index spécifié.

DataGrid.headerHeight

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDeDonnées.headerHeight
```

Description

Propriété : hauteur de la barre d'en-tête de la grille de données. La valeur par défaut est 20.

Exemple

L'exemple suivant permet de définir la position de défilement en haut de l'affichage :

```
maGrilleDeDonnées.headerHeight = 30;
```

DataGrid.headerRelease

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
objetDécoute.headerRelease = function(objetEvt){
  // insérez votre code ici
}
occurrenceMaGrilleDeDonnées.addEventListener("headerRelease", objetDécoute)
```

Description

Evénement : diffusé à l'ensemble des écouteurs enregistrés lorsque l'utilisateur désélectionne un en-tête de colonne. Vous pouvez utiliser cet événement avec la propriété `DataGridColumn.sortOnHeaderRelease` pour empêcher tout tri automatique et vous permettre d'effectuer le tri selon vos préférences.

Les composants V2 utilisent un modèle d'événement dispatcher/écouteur. Lorsque le composant DataGrid distribue un événement `headerRelease`, cet événement est géré par une fonction (également appelée *gestionnaire*) associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`objetEvt`*) au gestionnaire. 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. L'objet événement de l'événement `DataGrid.headerRelease` possède deux propriétés supplémentaires :

`columnIndex` Nombre indiquant l'index de la colonne cible.

`type` La chaîne `"headerRelease"`.

Pour plus d'informations, consultez *Objets événement*, page 592.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `monEcouteur` est défini puis transmis à la méthode `grid.addEventListener()` comme second paramètre. L'objet événement est capturé par le gestionnaire `headerRelease` dans le paramètre *`objetEvt`*. Lorsque l'événement `headerRelease` est diffusé, une instruction `trace` est envoyée vers le panneau de sortie, comme suit :

```
var monEcouteur = new Object();
monEcouteur.headerRelease = function(event) {
    trace("L'en-tête de colonne " + event.columnIndex + " a été activé");
};
grid.addEventListener("headerRelease", monEcouteur);
```

DataGrid.hScrollPolicy

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDeDonnées.hScrollPolicy
```

Description

Propriété : spécifie si la grille de données comprend une barre de défilement horizontale. Cette propriété comporte trois valeurs : `"on"`, `"off"` et `"auto"`. La valeur par défaut est `"off"`.

Si vous avez défini `hScrollPolicy` sur `"off"`, les colonnes sont redimensionnées proportionnellement pour être adaptées à la largeur déterminée.

Exemple

L'exemple suivant définit la régulation de défilement horizontal sur automatique :

```
maGrilleDonnées.hScrollPolicy = "auto";
```

DataGrid.removeAllColumns()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDonnées.removeAllColumns()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : supprime tous les objets DataGridColumn de la grille de données. L'appel de cette méthode n'a aucun effet sur le fournisseur de données.

Exemple

L'exemple suivant supprime tous les objets DataGridColumn de `maGrilleDonnées` :

```
maGrilleDonnées.removeAllColumns();
```

DataGrid.removeColumnAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDonnées.removeColumnAt(index)
```

Paramètres

index Index de la colonne à supprimer.

Renvoie

Référence à l'objet DataGridColumn supprimé.

Description

Méthode : supprime l'objet DataGridColumn à l'index spécifié.

Exemple

L'exemple suivant supprime l'objet DataGridColumn à l'index 2 dans `maGrilleDonnées` :

```
maGrilleDonnées.removeColumnAt(2);
```

DataGrid.replaceItemAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDonnées.replaceItemAt(index, item)
```

Paramètres

index Index de l'élément à remplacer.

item Objet correspondant à la valeur de l'élément à utiliser comme valeur de remplacement.

Renvoie

La valeur précédente.

Description

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

Exemple

L'exemple suivant remplace l'élément à l'index 4 par l'élément défini dans `uneNouvelleValeur` :

```
var uneNouvelleValeur = {name:"Jean", value:"fatigué"};
var valPréc = maGrille.replaceItemAt(4, uneNouvelleValeur);
```

DataGrid.resizableColumns

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDonnées.resizableColumns
```

Description

Propriété : une valeur booléenne indiquant si les colonnes de la grille peuvent être étirées (`true`) ou non (`false`) par l'utilisateur. Cette propriété doit être définie sur `true` pour que les colonnes individuelles puissent être redimensionnées. La valeur par défaut est `true`.

Exemple

L'exemple suivant permet d'empêcher les utilisateurs de redimensionner les colonnes :

```
maGrilleDonnées.resizableColumns = false;
```

DataGrid.selectable

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDonnées.selectable
```

Description

Propriété : une valeur booléenne indiquant si l'utilisateur peut sélectionner (`true`) ou non (`false`) la grille de données. La valeur par défaut est `true`.

Exemple

L'exemple suivant permet d'empêcher l'utilisateur de sélectionner la grille

```
maGrilleDonnées.selectable = false;
```

DataGrid.showHeaders

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDonnées.showHeaders
```

Description

Propriété : une valeur booléenne indiquant si la grille de données doit afficher (`true`) ou non (`false`) les en-têtes de colonnes. Les en-têtes de colonnes, gris, se distinguent facilement des autres lignes d'une grille. Si `DataGrid.sortableColumns` est défini sur `true`, l'utilisateur peut trier le contenu d'une colonne en cliquant sur son en-tête. La valeur par défaut est `true`.

Exemple

L'exemple suivant masque les en-têtes de colonnes :

```
maGrilleDonnées.showHeaders = false;
```

Voir aussi

`DataGrid.sortableColumns`

DataGrid.sortableColumns

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDonnées.sortableColumns
```

Description

Propriété : une valeur booléenne indiquant si les colonnes de la grille de données peuvent être triées (`true`) ou non (`false`), lorsque l'utilisateur clique sur les en-têtes de colonne. Cette propriété doit être définie sur `true` pour que les colonnes individuelles puissent être triées. Cette propriété doit être définie sur `true` pour que l'événement `headerRelease` puisse être diffusé. La valeur par défaut est `true`.

Exemple

L'exemple suivant désactive le tri :

```
maGrilleDonnées.sortableColumns = false;
```

Voir aussi

`DataGrid.headerRelease`

DataGrid.spaceColumnsEqually()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDonnées.spaceColumnsEqually()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : redimensionne les colonnes de façon égale.

Exemple

L'exemple suivant redimensionne les colonnes de `maGrille` lorsque l'utilisateur clique sur l'un des en-têtes de colonnes :

```
maGrille.showHeaders = true
maGrille.dataProvider = [{guitar:"Flying V", name:"maggot"}, {guitar:"SG",
  name:"dreschie"}, {guitar:"jagstang", name:"vitapup"}];
gridLO = new Object();
gridLO.headerRelease = function(){
  maGrille.spaceColumnsEqually();
}
maGrille.addEventListener("headerRelease", gridLO);
```

Classe DataGridColumn (Flash Professionnel uniquement)

Nom de classe ActionScript mx.controls.gridclassesDataGridColumn

Vous pouvez créer et configurer des objets DataGridColumn à utiliser en tant que colonnes d'une grille de données. Un grand nombre de méthodes de la classe DataGrid est consacré à la gestion des objets DataGridColumn. Dans la grille de données, les objets DataGridColumn sont triés dans un tableau basé sur zéro ; 0 correspond à la colonne la plus à gauche. Une fois les colonnes ajoutées ou créées, vous pouvez appeler `DataGrid.getColumnAt(`*`index`*`)` pour y accéder.

Il existe trois manières d'ajouter ou de créer des colonnes dans une grille. Si vous souhaitez configurer les colonnes, il est préférable d'utiliser la deuxième ou la troisième méthode avant d'ajouter des données dans une grille. Ainsi vous n'aurez pas à créer les colonnes deux fois.

- L'ajout d'un DataProvider ou d'un élément comprenant plusieurs champs à une grille qui n'a pas d'objet DataGridColumn configuré génère automatiquement des colonnes pour chaque champ en ordre inverse de la boucle `for..in`.

- `DataGrid.columnNames` récupère les noms de champs des champs d'éléments souhaités et génère des objets DataGridColumn, dans l'ordre, pour chaque champ répertorié. Cette approche vous permet de sélectionner et de classer rapidement les colonnes, en faisant appel à un minimum de paramètres de configuration. Elle supprime toute information précédente liée à la colonne.
- La façon la plus souple d'ajouter des colonnes consiste à les précréer en tant qu'objets DataGridColumn et à les ajouter à la grille de données en utilisant `DataGrid.addColumn()`. Cette approche est très pratique car elle permet d'ajouter des colonnes de taille et de format adéquats alors que ces colonnes ne se trouvent pas encore dans la grille (le processeur est moins sollicité). Pour plus d'informations, consultez *Constructeur pour la classe DataGridColumn*, page 189.

Propriétés de la classe DataGridColumn

Propriété	Description
`DataGridColumn.cellRenderer`	L'identifiant de liaison d'un symbole à utiliser pour afficher les cellules dans cette colonne.
`DataGridColumn.columnName`	Lecture seule. Le nom du champ associé à la colonne.
`DataGridColumn.editable`	Une valeur booléenne indiquant si une colonne est modifiable (`true`) ou non (`false`).
`DataGridColumn.headerRenderer`	Le nom d'une classe à utiliser pour afficher l'en-tête de cette colonne.
`DataGridColumn.headerText`	Le texte de l'en-tête de cette colonne.
`DataGridColumn.labelFunction`	Fonction qui détermine le champ d'un élément à afficher.
`DataGridColumn.resizable`	Une valeur booléenne indiquant si une colonne peut être redimensionnée (`true`) ou non (`false`).
`DataGridColumn.sortable`	Une valeur booléenne indiquant si une colonne peut être triée (`true`) ou non (`false`).
`DataGridColumn.sortOnHeaderRelease`	Une valeur booléenne indiquant si une colonne peut être triée (`true`) ou non (`false`) lorsque l'utilisateur clique sur l'en-tête de cette dernière.
`DataGridColumn.width`	La largeur d'une colonne, en pixels.

Constructeur pour la classe DataGridColumn

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
new DataGridColumn(nom)
```

Paramètres

nom Chaîne indiquant le nom de l'objet DataGridColumn. Ce paramètre correspond au champ de chaque élément à afficher.

Renvoie

Rien.

Description

Constructeur : crée un objet DataGridColumn. Utilisez ce constructeur pour créer des colonnes à ajouter à un composant DataGrid. Une fois les objets DataGridColumn créés, vous pouvez les ajouter à une grille de données en appelant `DataGrid.addColumn()`.

Exemple

L'exemple suivant crée un objet DataGridColumn appelé `Emplacement` :

```
import mx.controls.gridclasses.DataGridColumn;
var colonne = new DataGridColumn("Emplacement");
```

DataGridColumn.cellRenderer

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDonnées.getColumnAt(index).cellRenderer
```

Description

Propriété : un identificateur de liaison pour un symbole à utiliser pour afficher des cellules dans cette colonne. Toutes les classes utilisées pour cette propriété doivent implémenter l'interface CellRenderer (consultez *API CellRenderer*, page 84.) La valeur par défaut est undefined.

Exemple

L'exemple suivant utilise un identificateur de liaison pour définir un nouvel objet cellRenderer :

```
maGrille.getColumnAt(3).cellRenderer = "MonCellRenderer";
```

DataGridColumn.columnName

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

maGrilleDonnées.getColumnAt(*index*).columnName

Description

Propriété (lecture seule) : le nom du champ associé à cette colonne. La valeur par défaut est le nom dont il est question dans le constructeur DataGridColumn.

Exemple

L'exemple suivant affecte le nom de la colonne située à la troisième position d'index à la variable `nom` :

```
var nom = maGrille.getColumnAt(3).columnName;
```

Voir aussi

Constructeur pour la classe DataGridColumn

DataGridColumn.editable

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

maGrilleDonnées.getColumnAt(*index*).editable

Description

Propriété : détermine si l'utilisateur peut modifier (`true`) ou non (`false`) la colonne. La propriété `DataGrid.editable` doit être définie sur `true` pour que les colonnes individuelles puissent être modifiées, même lorsque `DataGridColumn.editable` est défini sur `true`. La valeur par défaut est `true`.

Exemple

L'exemple suivant rend non modifiable la première colonne d'une grille :

```
maGrilleDonnées.getColumnAt(0).editable = false;
```

Voir aussi

`DataGrid.editable`

DataGridColumn.headerRenderer

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDonnées.getColumnAt(index).headerRenderer
```

Description

Propriété : une chaîne indiquant le nom de classe à utiliser pour afficher l'en-tête de cette colonne. Toutes les classes utilisées pour cette propriété doivent implémenter l'interface CellRenderer (consultez *API CellRenderer*, page 84). La valeur par défaut est undefined.

Exemple

L'exemple suivant utilise un identificateur de liaison pour définir un nouvel objet headerRenderer :

```
maGrille.getColumnAt(3).headerRenderer = "MonHeaderRenderer";
```

DataGridColumn.headerText

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDonnées.getColumnAt(index).headerText
```

Description

Propriété : le texte de l'en-tête de colonne. La valeur par défaut est le nom de la colonne.

Exemple

L'exemple suivant définit l'en-tête de colonne sur "Le prix" :

```
var maColonne = new DataGridColumn("prix");
maColonne.headerText = "Le prix";
```

DataGridColumn.labelFunction

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDonnées.getColumnAt(index).labelFunction
```

Description

Propriété : spécifie une fonction permettant de déterminer le champ (ou la combinaison de champs) de chaque élément à afficher. Cette fonction reçoit un paramètre, *item*, qui correspond à l'élément présenté et doit renvoyer une chaîne représentant le texte à afficher. Cette propriété peut être utilisée pour créer des colonnes virtuelles qui n'ont pas de champ équivalent dans l'élément.

Exemple

L'exemple suivant crée une colonne virtuelle :

```
var maCol = maGrille.addColumn("Soustotal");
maCol.labelFunction = function(item) {
  return "$" + (item.price + (item.price * salesTax));
};
```

DataGridColumn.resizable

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDonnées.getColumnAt(index).resizable
```

Description

Propriété : une valeur booléenne indiquant si l'utilisateur peut redimensionner (`true`) ou non (`false`) une colonne. La propriété `DataGrid.resizableColumns` doit être définie sur `true` pour que cette propriété prenne effet. La valeur par défaut est `true`.

Exemple

L'exemple suivant empêche l'utilisateur de redimensionner la colonne à l'index 1 :

```
maGrille.getColumnAt(1).resizable = false;
```

DataGridColumn.sortable

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

maGrilleDonnées.getColumnAt(*index*).sortable

Description

Propriété : une valeur booléenne indiquant si l'utilisateur peut trier (`true`) ou non (`false`) une colonne. La propriété `DataGrid.sortableColumns` doit être définie sur `true` pour que cette propriété prenne effet. La valeur par défaut est `true`.

Exemple

L'exemple suivant empêche l'utilisateur de trier la colonne à l'index 1 :

```
maGrille.getColumnAt(1).sortable = false;
```

DataGridColumn.sortOnHeaderRelease

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

maGrilleDonnées.getColumnAt(*index*).sortOnHeaderRelease

Description

Propriété : une valeur booléenne indiquant si la colonne peut être triée automatiquement (`true`) ou non (`false`) lorsque l'utilisateur clique sur un en-tête. Cette propriété peut uniquement être définie sur `true` si `DataGridColumn.sortable` est défini sur `true`. Si `DataGridColumn.sortOnHeaderRelease` est défini sur `false`, vous pouvez utiliser l'événement `headerRelease` et effectuer votre propre tri.

La valeur par défaut est `true`.

Exemple

L'exemple suivant vous permet d'utiliser l'événement `headerRelease` pour effectuer votre propre tri :

```
maGrille.getColumnAt(7).sortOnHeaderRelease = false;
```

DataGridColumn.width

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maGrilleDonnées.getColumnAt(index).width
```

Description

Propriété : un nombre qui indique la largeur d'une colonne, en pixels. La valeur par défaut est 50.

Exemple

L'exemple suivant divise par deux la valeur par défaut de la taille d'une colonne :

```
maGrille.getColumnAt(4).width = 25;
```

Composant DataHolder (Flash Professionnel uniquement)

Le composant DataHolder est un référentiel pour les données et un moyen de générer des événements lors de la modification de ces données. Il permet essentiellement de stocker les données et joue le rôle de connecteur entre d'autres composants, par l'intermédiaire de la liaison de données.

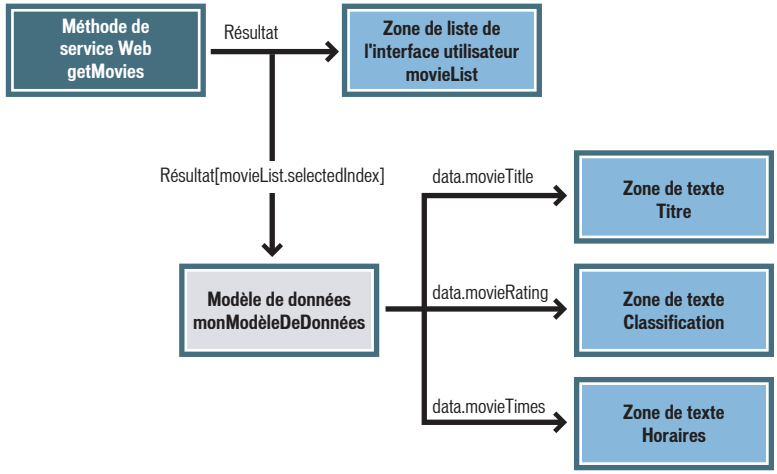
Le composant DataHolder possède initialement une seule propriété de liaison, appelée `data`. Vous pouvez ajouter davantage de propriétés via l'onglet Schéma du panneau Inspecteur de composants (Fenêtre > Panneaux de développement > Inspecteur de composants). Pour plus d'informations sur l'utilisation de l'onglet Schéma, consultez « Utilisation des schémas dans l'onglet Schéma (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Vous pouvez affecter tout type de données à une propriété DataHolder, soit en créant une liaison entre les données et une autre propriété, soit en utilisant votre propre code ActionScript. Lorsque les valeurs des données sont modifiées, le composant DataHolder émet un événement dont le nom est identique à celui de la propriété et exécute toute liaison associée à cette propriété.

Le composant DataHolder est particulièrement utile lorsqu'il est impossible d'associer directement des composants (tels que des connecteurs, des composants d'interface utilisateur ou des composants DataSet). Vous trouverez ci-dessous des scénarios dans lesquels vous pouvez utiliser un composant DataHolder :

- Si une valeur de données est générée par ActionScript, vous pouvez l'associer à d'autres composants. Dans ce cas, vous pouvez avoir un composant DataHolder qui contient des propriétés associées à votre convenance. Lorsque de nouvelles valeurs sont affectées à ces propriétés (à l'aide d'ActionScript, par exemple), ces valeurs sont distribuées à l'objet de liaison des données.

- Vous pouvez par exemple avoir une valeur de données résultant d'une liaison des données indexée et complexe, comme dans le diagramme suivant :



Dans ce cas, il est pratique de lier la valeur de données à un composant DataHolder (appelé *DataModel* dans cette illustration), puis de l'utiliser pour des liaisons avec l'interface utilisateur.

Création d'une application avec le composant DataHolder (Flash Professionnel uniquement)

Dans cet exemple, vous ajoutez une propriété Array à un schéma du composant DataHolder (un tableau) dont la valeur est déterminée par le code ActionScript que vous rédigez. Liez ensuite cette propriété Array à la propriété `dataProvider` d'un composant DataGrid, en utilisant l'onglet Liaisons du panneau Inspecteur de composants.

Pour utiliser le composant DataHolder dans une application simple :

1. Dans Flash MX Professionnel 2004, créez un nouveau fichier.
2. Ouvrez le panneau Composants (Fenêtre > Panneaux de développement > Composants), faites glisser un composant DataHolder sur la scène et nommez-le **dataHolder**.
3. Faites glisser un composant DataGrid sur la scène et nommez-le **grilleNoms**.
4. Sélectionnez le composant DataHolder, puis ouvrez le panneau Inspecteur de composants (Fenêtre > Panneaux de développement > Inspecteur de composants).
5. Dans le panneau Inspecteur de composants, cliquez sur l'onglet Schéma.
6. Cliquez sur le bouton Ajouter une propriété de composant (+), situé dans le panneau supérieur de l'onglet Schéma.
7. Dans le panneau inférieur de l'onglet Schéma, tapez **tableauNoms** dans le champ Nom du champ et sélectionnez Array dans le menu déroulant Types de données.
8. Dans le panneau Inspecteur de composants, cliquez sur l'onglet Liaisons et ajoutez une liaison entre la propriété `tableauNoms` du composant DataHolder et la propriété `dataProvider` du composant DataGrid.

   Pour plus d'informations sur la création de liaisons à partir de l'onglet Liaisons, consultez « Utilisation des liaisons dans l'onglet Liaisons (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

9 Dans le Scénario (Fenêtre > Scénario), sélectionnez la première image du Calque 1, puis ouvrez le panneau Actions (Fenêtre > Panneaux de développement > Actions).

10 Entrez le code suivant dans le panneau Actions :

```
dataHolder.tableauNoms= [{name:"Tom"},{name:"Paul"},{name:"Jean"}];
```

Ce code remplit le tableau `tableauNoms` avec plusieurs objets. Lorsque l'affectation de variable s'exécute, la liaison que vous avez établie précédemment entre le composant DataHolder et le composant DataGrid s'exécute.

11 Testez le fichier en sélectionnant Contrôle > Tester l'animation.

Propriétés de la classe DataHolder

Propriété	Description
`DataHolder.data`	Propriété de liaison par défaut du composant DataHolder.

DataHolder.data

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`dataHolder``.data`

Description

Propriété : l'élément par défaut dans un schéma d'objet DataHolder. Cette propriété n'est pas un membre « permanent » du composant DataHolder. Il s'agit plutôt de la propriété de liaison par défaut pour chaque occurrence du composant. Vous pouvez ajouter vos propres propriétés de liaison ou supprimer la propriété `data` par défaut, dans l'onglet Schéma du panneau Inspecteur de composants.

Pour plus d'informations sur l'utilisation de l'onglet Schéma, consultez « Utilisation des schémas dans l'onglet Schéma (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Exemple

Pour obtenir un exemple d'utilisation de ce composant, consultez *Création d'une application avec le composant DataHolder (Flash Professionnel uniquement)*, page 195.

API DataProvider

Nom de classe ActionScript mx.controls.listclasses.DataProvider

L'API DataProvider est un jeu de méthodes et de propriétés que doit posséder une source de données pour qu'une classe basée sur des listes puisse communiquer avec elle. Arrays, RecordSets et DataSet implémentent tous cette API. Vous pouvez créer une classe compatible DataProvider en implémentant toutes les méthodes et propriétés présentées dans ce manuel. Un composant basé sur des listes peut ensuite utiliser cette classe comme fournisseur de données.

Les méthodes de l'API DataProvider vous permettent d'interroger et de modifier les données de n'importe quel composant affichant des données (également appelé *affichage*). L'API DataProvider diffuse également des événements change lorsque les données sont modifiées. Plusieurs affichages peuvent utiliser le même fournisseur de données et recevoir tous les événements change.

Un fournisseur de données est un ensemble linéaire (comme un tableau) d'éléments. Chaque élément est un objet composé de plusieurs champs de données. Vous pouvez accéder à ces éléments via leur index (comme dans un tableau), en utilisant `DataProvider.getItemAt()`.

Les fournisseurs de données sont fréquemment utilisés dans les tableaux. Les composants de données appliquent toutes les méthodes de l'API DataProvider à `Array.prototype` lorsqu'un objet Array se trouve dans la même image ou dans le même écran qu'un composant de données. Cela vous permet d'utiliser tout tableau existant en tant que données pour les affichages qui ont une propriété `dataProvider`.

Du fait de l'API DataProvider, les composants v2 qui permettent l'affichage des données (DataGrid, List, Tree, etc.) peuvent également afficher Flash Remoting RecordSets et des données du composant DataSet. L'API DataProvider est le langage qu'utilisent les composants de données pour communiquer avec leurs fournisseurs de données.

Dans la documentation Macromedia Flash, « DataProvider » est le nom de l'API, `dataProvider` est une propriété de chaque composant qui agit comme un affichage pour les données, et « fournisseur de données » est le terme générique utilisé pour désigner une source de données.

Méthodes de l'API DataProvider

Nom	Description
`DataProvider.addItem()`	Ajoute un élément à la fin du fournisseur de données.
`DataProvider.addItemAt()`	Ajoute un élément au fournisseur de données à l'emplacement spécifié.
`DataProvider.editField()`	Modifie un champ du fournisseur de données.
`DataProvider.getEditingData()`	Obtient les données en vue d'une modification à partir d'un fournisseur de données.
`DataProvider.getItemAt()`	Obtient une référence à l'élément à l'emplacement spécifié.
`DataProvider.getItemID()`	Renvoie l'ID unique de l'élément.
`DataProvider.removeAll()`	Supprime tous les éléments du fournisseur de données.
`DataProvider.removeItemAt()`	Supprime un élément du fournisseur de données à l'emplacement spécifié.
`DataProvider.replaceItemAt()`	Remplace l'élément à l'emplacement spécifié par un autre élément.
`DataProvider.sortItems()`	Trie les éléments d'un fournisseur de données.
`DataProvider.sortItemsBy()`	Trie les éléments d'un fournisseur de données à l'aide de la fonction de comparaison spécifiée.

Propriétés de l'API DataProvider

Nom	Description
DataProvider.length	Le nombre d'éléments dans un fournisseur de données.

Evénements de l'API DataProvider

Nom	Description
DataProvider.modelChanged	Diffusé lorsque le fournisseur de données est modifié.

DataProvider.addItem()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

monFD.addItem(*item*)

Paramètres

item Un objet contenant des données. Y compris un élément d'un fournisseur de données.

Renvoie

Rien.

Description

Méthode : ajoute un nouvel élément à la fin du fournisseur de données.

Cette méthode déclenche l'événement `modelChanged` avec l'événement `addItems`.

Exemple

L'exemple suivant ajoute un élément à la fin du fournisseur de données `monFD` :

```
monFD.addItem({ label : "ceci est un élément"});
```

DataProvider.addItemAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

monFD.addItemAt(*index*, *item*)

Paramètres

index Nombre supérieur ou égal à 0. La position à laquelle insérer l'élément, l'index du nouvel élément.

item Un objet contenant les données de l'élément.

Renvoie

Rien.

Description

Méthode : ajoute un nouvel élément au fournisseur de données à l'index spécifié. Les index supérieurs à la longueur du fournisseur de données sont ignorés.

Cette méthode déclenche l'événement `modelChanged` avec l'événement `addItems`.

Exemple

L'exemple suivant ajoute un élément au fournisseur de données `monFD` à la quatrième place :

```
monFD.addItemAt(3, {label : "c'est le quatrième élément"});
```

DataProvider.editField()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monFD.editField(index, nomDeChamp, nouvellesDonnées)
```

Paramètres

index Nombre supérieur ou égal à 0. L'index de l'élément.

nomDeChamp Chaîne indiquant le nom du champ dans l'élément à modifier.

nouvellesDonnées Nouvelles données à placer dans le fournisseur de données.

Renvoie

Rien.

Description

Méthode : modifie un champ dans le fournisseur de données.

Cette méthode déclenche l'événement `modelChanged` avec l'événement `updateField`.

Exemple

Le code suivant modifie le champ `étiquette` du troisième élément :

```
monFD.editField(2, "étiquette", "mesNouvellesDonnées");
```

DataProvider.getEditingData()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monFD.getEditingData(index, nomDeChamp)
```

Paramètres

index Nombre supérieur ou égal à 0 et inférieur à *DataProvider.length*. Index de l'élément à récupérer.

nomDeChamp Chaîne indiquant le nom du champ qui est modifié.

Renvoie

Les données formatées modifiables à utiliser.

Description

Méthode : récupère les données en vue d'une modification à partir d'un fournisseur de données. Cela permet au modèle de données de proposer plusieurs formats de données pour modification et affichage.

Exemple

Le code suivant obtient une chaîne modifiable pour le champ prix :

```
trace(monFD.getEditingData(4, "prix");
```

DataProvider.getItemAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monFD.getItemAt(index)
```

Paramètres

index Nombre supérieur ou égal à 0 et inférieur à *DataProvider.length*. Index de l'élément à récupérer.

Renvoie

Une référence à l'élément récupéré, undefined si l'index est en dehors des limites.

Description

Méthode : récupère une référence à l'élément à un emplacement spécifié.

Exemple

Le code suivant affiche l'étiquette du cinquième élément :

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

DataProvider.getItemID()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004 Professionnel.

Usage

```
monFD.getItemID(index)
```

Paramètres

index Nombre supérieur ou égal à 0.

Renvoie

Un nombre correspondant à l'ID unique de l'élément.

Description

Méthode : renvoie un ID unique de l'élément. Cette méthode est principalement utilisée pour suivre la sélection. Cet ID est utilisé dans les composants de données pour conserver les listes des éléments sélectionnés.

Exemple

Cet exemple obtient l'ID du quatrième élément :

```
var ID = monFD.getItemID(3);
```

DataProvider.modelChanged

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
objetDécoute.modelChanged = function(objetEvt){
  // insérez votre code ici
}
monMenu.addEventListener("modelChanged", objetDécoute
```

Description

Evénement : diffusé à tous ses écouteurs d'affichage lorsque le fournisseur de données est modifié. Un écouteur est en général ajouté à un modèle en affectant sa propriété `dataProvider`.

Les composants V2 utilisent un modèle d'événement dispatcher/écouteur. Lorsqu'un fournisseur de données est modifié de quelque manière que ce soit, il diffuse un événement `modelChanged`, récupéré par les composants de données qui mettent à jour leurs affichages pour faire apparaître les modifications.

L'objet événement de l'événement `Menu.modelChanged` compte cinq propriétés supplémentaires :

- `eventName` La propriété `eventName` permet de séparer en sous-catégories des événements `modelChanged`. Les composants de données utilisent cette information pour éviter d'actualiser l'occurrence de composant (affichage) qui utilise le fournisseur de données. Les valeurs supportées de la propriété `eventName` sont les suivantes :
  - `updateAll` L'intégralité de la vue doit être actualisée, en excluant la position de défilement.
  - `addItems` Une série d'éléments a été ajoutée.
  - `removeItems` Une série d'éléments a été supprimée.
  - `updateItems` Une série d'éléments doit être actualisée.
  - `sort` Les données ont été triées.
  - `updateField` Un champ au sein d'un élément doit être modifié et actualisé.
  - `updateColumn` Une définition de champ complète au sein de dataProvider doit être actualisée.
  - `filterModel` Le modèle a été filtré et l'affichage doit être actualisé (réinitialiser scrollPosition).
  - `schemaLoaded` La définition de champ de dataProvider a été déclarée.
- `firstItem` L'index du premier élément affecté.
- `lastItem` L'index du dernier élément affecté. La valeur est égale à `firstItem` si seul un élément est affecté.
- `removedIDs` Un tableau des identificateurs d'éléments supprimés.
- `fieldName` Une chaîne indiquant le nom du champ affecté.

Pour plus d'informations sur les objets événement, consultez *Objets événement*, page 592.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `listener` est défini et transmis à la méthode `addEventListener()` en tant que second paramètre. L'objet événement est capturé par le gestionnaire `modelChanged` dans le paramètre `evt`. Lorsque l'événement `modelChanged` est diffusé, une instruction `trace` est envoyée vers le panneau de sortie, comme suit :

```
listener = new Object();
listener.modelChanged = function(evt){
    trace(evt.eventName);
}
maListe.addEventListener("modelChanged", listener);
```

DataProvider.length

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monFD.length
```

Description

Propriété (lecture seule) : nombre d'éléments dans le fournisseur de données.

Exemple

Cet exemple envoie le nombre d'éléments du fournisseur de données `monTableau` vers le panneau de sortie :

```
trace(monTableau.length);
```

DataProvider.removeAll()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monFD.removeAll()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : supprime tous les éléments du fournisseur de données.

Cette méthode déclenche l'événement `modelChanged` avec l'événement `removeItems`.

Exemple

Cet exemple supprime tous les éléments du fournisseur de données :

```
monFD.removeAll();
```

DataProvider.removeItemAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monFD.removeItemAt(index)
```

Paramètres

index Nombre supérieur ou égal à 0. L'index de l'élément à supprimer.

Renvoie

Rien.

Description

Méthode : supprime l'élément à l'emplacement d'index spécifié. Un index disparaît parmi les index situés après l'index supprimé.

Cette méthode déclenche l'événement `modelChanged` avec l'événement `removeItems`.

Exemple

Cet exemple supprime l'élément situé en quatrième position :

```
monFD.removeItemAt(3);
```

DataProvider.replaceItemAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monFD.replaceItemAt(index, item)
```

Paramètres

index Nombre supérieur ou égal à 0. L'index de l'élément à modifier.

item Un objet qui est le nouvel élément.

Renvoie

Rien.

Description

Méthode : remplace le contenu de l'élément à l'index spécifié.

Cette méthode déclenche l'événement `modelChanged` avec l'événement `removeItems`.

Exemple

Cet exemple remplace l'élément à l'index 3 par l'élément possédant l'étiquette "nouvelle étiquette" :

```
monFD.replaceItemAt(3, {label : "nouvelle étiquette"});
```

DataProvider.sortItems()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monFD.sortItems([compareFunc], [optionsFlag])
```

Paramètres

compareFunc Référence à une fonction utilisée pour comparer deux éléments, afin de déterminer leur ordre de tri. Pour plus d'informations, consultez `Array.sort()` dans le Dictionnaire ActionScript de l'aide. Ce paramètre est facultatif.

optionsFlag Permet d'effectuer plusieurs types de tri dans un seul tableau sans avoir à répliquer l'intégralité du tableau ou à le trier à plusieurs reprises. Ce paramètre est facultatif.

Les valeurs possibles pour *optionsFlag* sont les suivantes :

- `Array.DESCENDING`—trie par ordre décroissant.
- `Array.CASEINSENSITIVE`—trie sans respecter la casse.
- `Array.NUMERIC`—trie par ordre numérique si les deux éléments comparés sont des nombres. S'il ne s'agit pas de nombres, effectuez une comparaison de type chaîne (qui peut être non sensible à la casse si l'indicateur est spécifié).
- `Array.UNIQUESORT`—si deux objets du tableau sont identiques ou comportent des champs de tri identiques, cette méthode renvoie un code d'erreur (0) au lieu d'un tableau trié.
- `Array.RETURNINDEXEDARRAY`—renvoie un tableau d'index d'entiers correspondant au résultat du tri. Par exemple, si le tableau suivant est trié avec le paramètre *optionsFlag* contenant la valeur `Array.RETURNINDEXEDARRAY`, il renvoie la seconde ligne de code et le tableau reste inchangé :

  ```
  ["a", "d", "c", "b"]
  [0, 3, 2, 1]
  ```

Vous pouvez associer ces options au sein d'une valeur. Par exemple, le code suivant associe les options 3 et 1 :

```
array.sort (Array.NUMERIC | Array.DESCENDING)
```

Renvoie

Rien.

Description

Méthode : trie les éléments dans le fournisseur de données selon la fonction de comparaison spécifiée par le paramètre *compareFunc* ou selon une ou plusieurs options de tri spécifiées par le paramètre *optionsFlag*.

Cette méthode déclenche l'événement `modelChanged` avec l'événement `sort`.

Exemple

Cet exemple trie selon des étiquettes majuscules. Les éléments `a` et `b` sont transmis à la fonction et possèdent les champs `label` et `data` :

```
maListe.sortItems(upperCaseFunc);
function upperCaseFunc(a,b){
    return a.label.toUpperCase() > b.label.toUpperCase();
}
```

DataProvider.sortItemsBy()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monFD.sortItemsBy(nomDeChamp, ordre, [optionsFlag])
```

Paramètres

nomDeChamp Une chaîne spécifiant le nom du champ à utiliser pour le tri. Cette valeur est en général `"label"` ou `"data"`.

ordre Une chaîne spécifiant le tri des éléments par ordre croissant (`"ASC"`) ou décroissant (`"DESC"`).

optionsFlag Permet d'effectuer plusieurs types de tri dans un seul tableau sans avoir à répliquer l'intégralité du tableau ou à le trier à plusieurs reprises. Ce paramètre est facultatif.

Les valeurs possibles pour *optionsFlag* sont les suivantes :

- `Array.DESCENDING`—trie par ordre décroissant.
- `Array.CASEINSENSITIVE`—trie sans respecter la casse.
- `Array.NUMERIC`—trie par ordre numérique si les deux éléments comparés sont des nombres. S'il ne s'agit pas de nombres, effectuez une comparaison de type chaîne (qui peut être non sensible à la casse si l'indicateur est spécifié).
- `Array.UNIQUESORT`—si deux objets du tableau sont identiques ou comportent des champs de tri identiques, cette méthode renvoie un code d'erreur (0) au lieu d'un tableau trié.
- `Array.RETURNINDEXEDARRAY`—renvoie un tableau d'index d'entiers correspondant au résultat du tri. Par exemple, si le tableau suivant est trié avec le paramètre *optionsFlag* contenant la valeur `Array.RETURNINDEXEDARRAY`, il renverra la seconde ligne de code et le tableau restera inchangé :

  ```
  ["a", "d", "c", "b"]
  [0, 3, 2, 1]
  ```

Vous pouvez associer ces options au sein d'une valeur. Par exemple, le code suivant associe les options 3 et 1 :

```
array.sort (Array.NUMERIC | Array.DESCENDING)
```

Renvoie

Rien.

Description

Méthode : trie les éléments du fournisseur de données par ordre alphabétique ou numérique, dans l'ordre spécifié, avec le nom de champ 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 les programmeurs expérimentés peuvent spécifier n'importe quelle primitive. Vous pouvez également utiliser le paramètre *optionsFlag* pour spécifier un type de tri.

Cette méthode déclenche l'événement `modelChanged` avec l'événement `sort`.

Exemple

Le code suivant trie les éléments d'une liste par ordre croissant en utilisant leurs étiquettes.

```
monFD.sortItemsBy("label", "ASC");
```

Composant DataSet (Flash Professionnel uniquement)

Le composant DataSet vous permet d'utiliser les données en tant que collections d'objets pouvant être indexés, triés, recherchés, filtrés et modifiés.

La fonctionnalité du composant DataSet comprend DataSetIterator, jeu de méthodes pour parcourir et manipuler une collection de données, et DeltaPacket, jeu d'interfaces et de classes pour utiliser des mises à jour d'une collection de données. Dans la plupart des cas, vous n'utilisez pas directement ces classes et interfaces. Vous les utilisez indirectement par l'intermédiaire de méthodes fournies par la classe DataSet.

Les éléments gérés par le composant DataSet sont également appelés *objets de transfert*. Un objet de transfert expose les données métier qui résident sur le serveur avec des attributs publics ou des méthodes d'accesseur pour lire et écrire des données. Le composant DataSet permet aux développeurs d'utiliser des objets complexes côté client qui reflètent à l'identique les objets côté serveur ou, plus simplement, d'utiliser une collection d'objets anonymes, avec des attributs publics, représentant les champs d'un enregistrement de données. Pour plus d'informations sur les objets de transfert, consultez la page Core J2EE Patterns Transfer Object à l'adresse suivante : java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html.

Remarque : Le composant DataSet requiert Flash Player 7 ou version ultérieure.

Utilisation du composant DataSet (Flash Professionnel uniquement)

En règle générale, vous utilisez le composant DataSet dans une application avec d'autres composants pour manipuler et mettre à jour une source de données : un composant Connector pour la connexion vers une source de données externe, des composants d'interface utilisateur pour l'affichage des données en provenance de la source de données et un composant Resolver pour la conversion des mises à jour de l'ensemble de données au format approprié, pour l'envoi vers la source de données externe. Vous pouvez ensuite utiliser une liaison de données pour lier entre elles les propriétés de ces différents composants.

Pour plus d'informations sur le composant DataSet et sur son utilisation avec d'autres composants, consultez « Gestion des données (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Paramètres du composant DataSet

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

itemClassName Le nom de la classe d'objet de transfert instanciée chaque fois qu'un nouvel élément est nécessaire.

Remarque : Pour que la classe spécifiée soit disponible à l'exécution, vous devez également inclure une référence pleinement qualifiée à cette classe dans le code de votre fichier SWF (par exemple : `var monElément:my.package.monElément;`).

filtered Si la valeur est `true`, un filtre est appliqué à l'ensemble de données de sorte qu'il contienne uniquement les objets correspondant au critère défini dans le filtre.

logChanges Si la valeur est `true`, l'ensemble de données garde une trace de tous les changements (de données ou d'appels de méthodes) dans sa propriété `deltaPacket`.

readOnly Si la valeur est `true`, l'ensemble de données ne peut pas être modifié.

Vous pouvez utiliser ActionScript pour contrôler ces options ainsi que d'autres options du composant DataSet à l'aide des propriétés, méthodes et événements ActionScript. Pour plus d'informations, consultez *Classe DataSet (Flash Professionnel uniquement)*, page 210.

Création d'une application avec le composant DataSet

Généralement, vous utilisez le composant DataSet avec d'autres composants d'interface utilisateur, et souvent avec un composant Connector comme le composant XMLConnector ou WebServiceConnector. Les éléments de l'ensemble de données sont remplis via le composant Connector ou au moyen de données ActionScript brutes, puis liés aux contrôles de l'interface utilisateur (tels que les composants List ou DataGrid).

Pour créer une application avec le composant DataSet :

1. Dans Flash MX Professionnel 2004, Sélectionnez Fichier > Nouveau. Dans la colonne Type, sélectionnez Document Flash et cliquez sur OK.
2. Ouvrez le panneau Composants (Fenêtre > Panneaux de développement > Composants) s'il n'est pas déjà ouvert.
3. Faites glisser un composant DataSet du panneau Composants jusqu'à la scène. Dans l'inspecteur des propriétés, nommez-le **donnéesUtilisateur**.
4. Faites glisser un composant DataGrid jusqu'à la scène et nommez-le **grilleUtilisateur**.
5. Redimensionnez le composant DataGrid pour qu'il compte environ 300 pixels de large et 100 pixels de haut.
6. Faites glisser un composant Button jusqu'à la scène et nommez-le **btnSuiv**.
7. Dans le scénario, sélectionnez la première image dans le calque 1 et ouvrez le panneau Actions (Fenêtre > Panneaux de développement > Actions).

8 Ajoutez le code suivant au panneau Actions

```
var recData = [{id:0, firstName:"Michel", lastName:"Jourdain"},
               {id:1, firstName:"Jean", lastName:"Sadourni"},
               {id:2, firstName:"Paul", lastName:"Simon"}];
donnéesUtilisateur.items = recData;
```

Cela remplit la propriété `items` de l'objet DataSet avec un tableau d'objets, chacun d'entre eux possédant trois propriétés : `firstName`, `lastName` et `id`.

9 Pour lier le contenu du composant DataSet au contenu du composant DataGrid, ouvrez le panneau Inspecteur de composants (Fenêtre > Panneaux de développement > Inspecteur de composants) et cliquez sur l'onglet Liaisons.

10 Sélectionnez le composant DataGrid (`grilleUtilisateur`) sur la scène et cliquez sur le bouton Ajouter une liaison (+) dans le panneau Inspecteur de composants.

11 Dans la boîte de dialogue Ajouter une liaison, sélectionnez « dataProvider : Array » et cliquez sur OK.

12 Dans le panneau Inspecteur de composants, double-cliquez sur le champ bound to.

13 Dans la boîte de dialogue Lié à qui apparaît, sélectionnez « Ensemble de données <donnéesUtilisateur> » dans la colonne Chemin du composant, puis sélectionnez « dataProvider : Array » dans la colonne Emplacement du schéma.

14 Pour lier l'index sélectionné du composant DataSet à l'index sélectionné du composant DataGrid, cliquez à nouveau sur le bouton Ajouter une liaison (+) dans le panneau Inspecteur de composants.

15 Dans la boîte de dialogue qui apparaît, sélectionnez « selectedIndex : Number ». Cliquez sur OK.

16 Pour ouvrir la boîte de dialogue Lié à, double-cliquez sur le champ bound to dans le panneau Inspecteur de composants.

17 Dans le champ Chemin du composant, sélectionnez « Ensemble de données <donnéesUtilisateur> » dans la colonne Chemin du composant et sélectionnez « selectedIndex : Number » dans la colonne Emplacement du schéma.

18 Sélectionnez le composant Button (`btnSuiv`) et ouvrez le panneau Actions (Fenêtre > Panneaux de développement > Actions), si ce dernier n'est pas déjà ouvert.

19 Entrez le code suivant dans le panneau Actions :

```
on(click) {
  _parent.userData.next();
}
```

Ce code utilise la méthode `DataSet.next()` pour naviguer vers l'élément suivant dans la collection d'éléments de l'objet DataSet. Comme vous avez lié la propriété `selectedIndex` de l'objet DataGrid à la même propriété de l'objet DataSet, une modification apportée à l'élément courant dans l'objet DataSet se répercute également sur l'élément courant (sélectionné) dans l'objet DataGrid.

20 Enregistrez le fichier, puis, pour tester le fichier SWF, sélectionnez Contrôle > Tester l'animation.

L'objet DataGrid est rempli avec les éléments spécifiés. Notez que lorsque vous cliquez sur le bouton, l'élément sélectionné dans l'objet DataGrid change.

Classe DataSet (Flash Professionnel uniquement)

Nom de classe ActionScript mx.data.components.DataSet

Méthodes de la classe DataSet

Méthode	Description
DataSet.addItem()	Ajoute l'élément spécifié à la collection.
DataSet.addSort()	Crée un nouvel affichage trié des éléments de la collection.
DataSet.applyUpdates()	Indique aux écouteurs que les modifications de l'objet DataSet sont prêtes.
DataSet.changesPending()	Indique s'il y a des éléments dans l'objet DeltaPacket.
DataSet.clear()	Efface tous les éléments de l'affichage courant de la collection.
DataSet.createItem()	Renvoie un élément de collection nouvellement initialisé.
DataSet.disableEvents()	Arrête l'envoi d'événements DataSet aux écouteurs.
DataSet.enableEvents()	Relance l'envoi d'événements DataSet aux écouteurs.
DataSet.find()	Localise un élément dans l'affichage courant de la collection.
DataSet.findFirst()	Localise la première occurrence d'un élément dans l'affichage courant de la collection.
DataSet.findLast()	Localise la dernière occurrence d'un élément dans l'affichage courant de la collection.
DataSet.first()	Effectue un déplacement vers le premier élément dans l'affichage courant de la collection.
DataSet.getItemId()	Renvoie l'ID unique de l'élément spécifié.
DataSet.getIterator()	Renvoie un clone de l'itérateur courant.
DataSet.hasNext()	Indique si l'itérateur courant est parvenu à la fin de l'affichage de la collection.
DataSet.hasPrevious()	Indique si l'itérateur courant est parvenu au début de l'affichage de la collection.
DataSet.hasSort()	Indique si le tri spécifié existe.
DataSet.isEmpty()	Indique si la collection contient des éléments.
DataSet.last()	Déplace le dernier élément dans l'affichage courant de la collection.
DataSet.loadFromSharedObj()	Récupère le contenu d'un objet DataSet à partir d'un objet partagé.
DataSet.locateById()	Déplace l'itérateur courant vers l'élément possédant l'ID spécifié.
DataSet.next()	Effectue un déplacement vers l'élément suivant dans l'affichage courant de la collection.
DataSet.previous()	Effectue un déplacement vers l'élément précédent dans l'affichage courant de la collection.
DataSet.removeAll()	Supprime tous les éléments de la collection.

Méthode	Description
DataSet.removeItem()	Supprime l'élément spécifié de la collection.
DataSet.removeRange()	Supprime les paramètres d'étendue de l'itérateur courant.
DataSet.removeSort()	Supprime le tri spécifié de l'objet DataSet.
DataSet.saveToSharedObj()	Enregistre les données de l'objet DataSet dans un objet partagé.
DataSet.setIterator()	Définit l'itérateur courant pour l'objet DataSet.
DataSet.setRange()	Définit les paramètres d'étendue de l'itérateur courant.
DataSet.skip()	Effectue un déplacement vers l'avant ou vers l'arrière selon un nombre spécifié d'éléments dans l'affichage courant de la collection.
DataSet.useSort()	Le tri spécifié devient actif.

Propriétés de la classe DataSet

Propriété	Description
DataSet.currentItem	Renvoie l'élément courant dans la collection.
DataSet.dataProvider	Renvoie l'interface DataProvider.
DataSet.deltaPacket	Renvoie les modifications apportées à la collection ou affecte des modifications à apporter à la collection.
DataSet.filtered	Indique si les éléments sont filtrés.
DataSet.filterFunc	Fonction définie par l'utilisateur pour filtrer des éléments dans la collection.
DataSet.items	Eléments de la collection.
DataSet.itemClassName	Objet à créer lors de l'affectation des éléments.
DataSet.length	Spécifie le nombre d'éléments dans l'affichage courant de la collection.
DataSet.logChanges	Indique si les modifications apportées à la collection ou à ses éléments sont enregistrées.
DataSet.properties	Contient les propriétés (champs) pour tout objet de transfert au sein de cette collection.
DataSet.readOnly	Indique si la collection peut être modifiée.
DataSet.schema	Spécifie le schéma de la collection au format XML.
DataSet.selectedIndex	Contient l'index de l'élément courant au sein de la collection.

Evénements de la classe DataSet

Evénement	Description
DataSet.addItem	Diffusé avant l'ajout d'un élément à la collection.
DataSet.afterLoaded	Diffusé après l'affectation de la propriété `items`.

Evénement	Description
DataSet.deltaPacketChanged	Diffusé lorsque le DeltaPacket de l'objet DataSet a été modifié et qu'il est prêt à être utilisé.
DataSet.calcFields	Diffusé lorsque les champs calculés doivent être mis à jour.
DataSet.iteratorScrolled	Diffusé lorsque la position de l'itérateur est modifiée.
DataSet.modelChanged	Diffusé lorsque les éléments de la collection ont été modifiés de quelque manière que ce soit.
DataSet.newItem	Diffusé lorsqu'un nouvel élément est construit par l'objet DataSet, mais avant son ajout à la collection.
DataSet.removeItem	Diffusé avant la suppression d'un élément.
DataSet.resolveDelta	Diffusé lorsqu'un objet DeltaPacket est affecté à l'objet DataSet qui contient des messages.

DataSet.addItem

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
on(addItem) {
  // insérez votre code ici
}
objetDécoute = new Object();
objetDécoute.addItem = function (objEvt) {
  // insérez votre code ici
}
dataSet.addEventListener("addItem", objetDécoute)
```

Description

Evénement : généré juste avant l'insertion d'un nouvel objet de transfert dans cette collection.

Si vous définissez la propriété `result` de l'objet événement sur `false`, l'opération d'ajout est annulée. Si vous la définissez sur `true`, l'opération d'ajout est autorisée.

L'objet événement (*`objEvt`*) possède les propriétés suivantes :

`target` L'objet DataSet qui a généré l'événement.

`type` La chaîne `"addItem"`.

`item` Une référence à l'élément de la collection à ajouter.

`result` Une valeur booléenne indiquant si l'élément spécifié doit être ajouté. Par défaut, cette valeur est `true`.

Exemple

Le gestionnaire d'événement `on(addItem)` suivant (associé à un objet DataSet) annule l'ajout du nouvel élément si une fonction, appelée `userHasAdminPrivs()`, définie par l'utilisateur, renvoie la valeur `false` ; sinon l'ajout de l'élément est autorisé.

```
on(addItem) {
  if(globalObj.userHasAdminPrivs()) {
    // Autoriser l'ajout de l'élément.
    objEvt.result = true;
} else {
    // Ne pas autoriser l'ajout de l'élément. L'utilisateur ne dispose pas des
  droits d'administrateur.
    objEvt.result = false;
  }
}
```

Voir aussi

`DataSet.removeItem`

DataSet.addItem()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

`dataSet``.addItem([`*`obj`*`])`

Paramètres

`obj` Un objet à ajouter à cette collection. Ce paramètre est facultatif.

Renvoie

Renvoie `true` si l'élément a été ajouté à la collection ; sinon, renvoie `false`.

Description

Méthode : ajoute l'objet de transfert spécifié à la collection pour gestion. L'élément qui vient d'être ajouté devient l'élément courant de l'ensemble de données. Si aucun paramètre *`obj`* n'est spécifié, un nouvel objet est créé automatiquement au moyen de `DataSet.createItem()`.

L'emplacement du nouvel élément dans la collection varie selon qu'un tri a été spécifié pour l'itérateur courant. Si aucun tri n'est défini, l'élément spécifié est ajouté à la fin de la collection. Si un tri est défini, l'élément est ajouté à la collection en fonction de sa position dans le tri courant.

Pour plus d'informations sur l'initialisation et la construction de l'objet de transfert, consultez `DataSet.createItem()`.

Exemple

```
monEnsembleDeDonnées.addItem(monEnsembleDeDonnées.createItem());
```

Voir aussi

`DataSet.createItem()`

DataSet.addSort()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

dataSet.addSort(*nom*, *listeChamps*, *optionsTri*)

Paramètres

nom Une chaîne spécifiant le nom du tri.

listeChamps Un tableau de chaînes spécifiant les noms des champs à trier.

optionsTri Une ou plusieurs des valeurs entières (constantes) suivantes, indiquant les options choisies pour ce tri. Sépare plusieurs valeurs en utilisant l'opérateur OR (|) au niveau du bit. Ces valeurs peuvent être :

- `DataSetIterator.Ascending` Trie les éléments par ordre croissant. Il s'agit de l'option de tri par défaut lorsque aucune option de tri n'est spécifiée.
- `DataSetIterator.Descending` Trie les éléments par ordre décroissant en fonction des propriétés d'éléments spécifiées.
- `DataSetIterator.Unique` Empêche le tri si des champs ont des valeurs similaires.
- `DataSetIterator.CaseInsensitive` Ignore la casse lors de la comparaison de deux chaînes, au cours du tri. Par défaut, le tri est sensible à la casse lorsque la propriété triée est une chaîne.

Une exception `DataSetError` est émise lorsque `DataSetIterator.Unique` est spécifié en tant qu'option de tri et que les données triées ne sont pas uniques, lorsque le nom de tri spécifié a déjà été ajouté ou lorsqu'une propriété spécifiée dans le tableau *listeChamps* n'existe pas dans cet ensemble de données.

Renvoie

Rien.

Description

Méthode : crée un nouveau tri croissant ou décroissant pour l'itérateur courant basé sur les propriétés spécifiées par le paramètre *listeChamps*. Le nouveau tri est automatiquement affecté à l'itérateur courant après sa création et son stockage dans la collection de tri pour une utilisation ultérieure.

Exemple

Le code suivant crée un nouveau tri, appelé `rank`. Celui -ci réalise un tri unique par ordre décroissant, sensible à la casse, sur le champ `classRank` de l'objet DataSet.

```
monEnsembleDeDonnées.addSort("rank", ["classRank"], DataSetIterator.Descending
  | DataSetIterator.Unique | DataSetIterator.CaseInsensitive);
```

Voir aussi

`DataSet.removeSort()`

DataSet.afterLoaded

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
on(afterLoaded) {
  // insérez votre code ici
}
objetDécoute = new Object();
objetDécoute.afterLoaded = function (objEvt) {
  // insérez votre code ici
}
dataSet.addEventListener("afterLoaded", objetDécoute)
```

Description

Evénement : diffusé immédiatement après l'affectation de la propriété `DataSet.items`.

L'objet événement (*objEvt*) possède les propriétés suivantes :

`target` L'objet DataSet qui a généré l'événement.

`type` La chaîne `afterLoaded`.

Exemple

Dans cet exemple, un formulaire appelé `formContact` (masqué) devient visible une fois que les éléments dans DataSet `contact_ds` ont été affectés.

```
contact_ds.addEventListener("afterLoaded", loadListener);
loadListener = new Object();
loadListener.afterLoaded = function (objEvt) {
  if(objEvt.target == "contact_ds") {
    formContact.visible = true;
  }
}
```

DataSet.applyUpdates()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
dataSet.applyUpdates()
```

Renvoie

Rien.

Description

Méthode : signale que la propriété `DataSet.deltaPacket` possède une valeur à laquelle vous pouvez accéder en utilisant la liaison de données ou directement via ActionScript. Avant l'appel de cette méthode, la propriété `DataSet.deltaPacket` est `null`. Cette méthode n'a aucun effet si des événements ont été désactivés au moyen de la méthode `DataSet.disableEvents()`.

L'appel de cette méthode crée également un ID de transaction pour la propriété `DataSet.deltaPacket` courante et émet un événement `deltaPacketChanged`. Pour plus d'informations, consultez `DataSet.deltaPacket`.

Exemple

Le code suivant appelle la méthode `applyUpdates()` sur `monEnsembleDeDonnées`.

```
monEnsembleDeDonnées.applyUpdates();
```

Voir aussi

`DataSet.deltaPacket`

DataSet.calcFields

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
on(calcFields) {
  // insérez votre code ici
}
objetDécoute = new Object();
objetDécoute.calcFields = function (objEvt) {
  // insérez votre code ici
}
dataSet.addEventListener("calcFields", objetDécoute)
```

Description

Evénement : généré lorsque des valeurs des champs calculés pour l'élément courant dans la collection doivent être déterminées. Un champ calculé est un champ dont la propriété Kind est définie sur Calculated dans l'onglet Schéma du panneau Inspecteur de composants. L'écouteur d'événement `calcFields` que vous créez doit effectuer le calcul requis et définir la valeur du champ calculé.

Cet événement est également appelé lorsque la valeur d'un champ non calculé (champ dont la propriété Kind est définie sur Data dans l'onglet Schéma du panneau Inspecteur de composants) est mise à jour.

Pour plus d'informations sur la propriété Kind, consultez « Types de schéma (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Attention : Ne modifiez pas les valeurs des champs non calculés dans cet événement car cela risque de provoquer une « boucle sans fin ». Définissez uniquement les valeurs des champs calculés au sein de l'événement `calcFields`.

DataSet.changesPending()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
dataSet.changesPending()
```

Renvoie

Une valeur booléenne.

Description

Méthode : renvoie `true` si la collection ou n'importe quel élément de la collection a des modifications en attente et que ces dernières n'ont pas encore été envoyées dans un objet DeltaPacket ; sinon, renvoie `false`.

Exemple

Le code suivant active un bouton Enregistrer les changements (masqué) si la collection DataSet, ou n'importe quel élément de cette collection, a fait l'objet de modifications qui n'ont pas été implémentées dans un objet DeltaPacket.

```
if( data_ds.changesPending() ) {
  EnregistrerChangements_btn.enabled = true;
}
```

DataSet.clear()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

dataSet.clear()

Renvoie

Rien.

Description

Méthode : supprime les éléments de l'affichage courant de la collection. Les éléments qui peuvent être affichés dépendent des paramètres de filtre et d'étendue courants de l'itérateur actuel. Par conséquent, l'appel de cette méthode risque de ne pas supprimer tous les éléments de la collection. Pour supprimer tous les éléments de la collection quel que soit l'affichage de l'itérateur actuel, utilisez DataSet.removeAll().

Si DataSet.logChanges est défini sur true lorsque vous invoquez cette méthode, les entrées « supprimées » sont ajoutées à DataSet.deltaPacket pour tous les éléments de la collection.

Exemple

Cet exemple supprime tous les éléments de l'affichage courant de la collection DataSet. La propriété logChanges étant définie sur true, la suppression de ces éléments est enregistrée.

```
monEnsembleDeDonnées.logChanges= true;
monEnsembleDeDonnées.clear();
```

Voir aussi

DataSet.deltaPacket, DataSet.logChanges

DataSet.createItem()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

dataSet.createItem([*itemData*])

Paramètres

itemData Données associées à l'élément. Ce paramètre est facultatif.

Renvoie

L'élément qui vient d'être construit.

Description

Méthode : crée un élément qui n'est pas associé à la collection. Vous pouvez spécifier la classe de l'objet créée avec la propriété `DataSet.itemClassName`. Si aucune valeur `DataSet.itemClassName` n'est spécifiée et que le paramètre *`itemData`* est omis, un objet anonyme est construit. Les propriétés de cet objet anonyme sont définies aux valeurs par défaut à partir du schéma actuellement spécifié par `DataSet.schema`.

Lorsque cette méthode est invoquée, tous les écouteurs de l'événement `DataSet.newItem` sont notifiés et en mesure de manipuler l'élément avant qu'il ne soit renvoyé par cette méthode. Les données d'éléments facultatives spécifiées sont utilisées pour initialiser la classe spécifiée avec la propriété `DataSet.itemClassName` ou comme élément si `DataSet.itemClassName` est vide.

Une exception `DataSetError` est émise lorsque la classe spécifiée avec la propriété `DataSet.itemClassName` ne peut pas être chargée.

Exemple

```
contact.itemClassName = "Contact";
var itemData = new XML("<contact_info><name>John Smith</
  name><phone>555.555.4567</phone><zip><pre>94025</pre><post>0556</post></
  zip></contact_info>");
contact.addItem(contact.createItem(itemData));
```

Voir aussi

`DataSet.itemClassName`, `DataSet.newItem`, `DataSet.schema`

DataSet.currentItem

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

`dataSet``.currentItem`

Description

Propriété (lecture seule) : renvoie l'élément courant dans la collection DataSet ou `null` si la collection est vide ou si l'affichage de la collection de l'itérateur courant est vide.

Cette propriété offre un accès direct à l'élément au sein de la collection. Les modifications apportées en accédant directement à l'objet ne sont pas suivies (dans la propriété `DataSet.deltaPacket`) et les paramètres de schéma ne sont pas appliqués aux propriétés de cet objet.

Exemple

L'exemple suivant affiche la valeur de la propriété `customerName` définie dans l'élément courant dans l'ensemble de données appelé `customerData`.

```
trace(customerData.currentItem.customerName);
```

DataSet.dataProvider

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

dataSet.dataProvider

Description

Propriété : l'interface DataProvider pour cet ensemble de données. Cette propriété fournit des données aux contrôles de l'interface utilisateur, comme les composants List et DataGrid.

Exemple

Le code suivant affecte la propriété `dataProvider` d'un objet DataSet à la propriété correspondante d'un composant DataGrid.

```
maGrille.dataProvider = monEnsembleDeDonnées.dataProvider;
```

DataSet.deltaPacket

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

dataSet.deltaPacket

Description

Propriété : renvoie un objet DeltaPacket contenant toutes les opérations de modification réalisées sur la collection *dataSet* et ses éléments. Cette propriété est `null` tant que `DataSet.applyUpdates()` est appelée sur *dataSet*.

Lorsque `DataSet.applyUpdates()` est appelée, un ID de transaction est affecté à l'objet DeltaPacket. Cet ID de transaction permet d'identifier l'objet DeltaPacket sur une boucle de mise à jour à partir du serveur vers le client. Toute affectation ultérieure à la propriété `deltaPacket` via un objet DeltaPacket possédant un ID de transaction correspondant est considérée comme étant la réponse du serveur aux modifications envoyées précédemment. Un objet DeltaPacket possédant un ID correspondant est utilisé pour mettre à jour la collection et rapporter les erreurs spécifiées au sein du paquet.

Les erreurs et les messages du serveur sont rapportés aux écouteurs de l'événement `DataSet.resolveDelta`. Notez que les paramètres `DataSet.logChanges` sont ignorés lorsqu'un objet DeltaPacket possédant un ID correspondant est affecté à `DataSet.deltaPacket`. Un objet DeltaPacket sans ID de transaction correspondant met à jour la collection, comme si les API DataSet étaient utilisées directement. Cela peut créer des entrées delta supplémentaires, en fonction du paramètre `DataSet.logChanges` courant de *dataSet* et de l'objet DeltaPacket.

Une exception `DataSetError` est émise si un objet DeltaPacket est affecté avec un ID de transaction correspondant et que l'un des éléments de l'objet DeltaPacket nouvellement affecté est introuvable dans l'objet DeltaPacket d'origine.

Voir aussi

`DataSet.applyUpdates()`, `DataSet.logChanges`, `DataSet.resolveDelta`

DataSet.deltaPacketChanged

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
on(deltaPacketChanged) {
  // insérez votre code ici
}
objetDécoute = new Object();
objetDécoute.deltaPacketChanged = function (objEvt) {
  // insérez votre code ici
}
dataSet.addEventListener("deltaPacketChanged", objetDécoute)
```

Description

Evénement : diffusé lorsque la propriété `deltaPacket` de l'objet DataSet spécifié a été modifiée et qu'elle est prête à être utilisée.

Voir aussi

`DataSet.deltaPacket`

DataSet.disableEvents()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
dataSet.disableEvents()
```

Renvoie

Rien.

Description

Méthode : désactive des événements pour l'objet DataSet. Lorsque des événements sont désactivés, aucun contrôle de l'interface utilisateur (tel qu'un composant DataGrid) n'est mis à jour lorsque des éléments de la collection sont modifiés ou lorsque l'objet DataSet atteint un nouvel élément de la collection.

Pour réactiver des événements, vous devez appeler `DataSet.enableEvents()`. Pour réactiver la distribution d'événements, la méthode `disableEvents()` peut être appelée plusieurs fois, mais `enableEvents()` doit être appelée autant de fois.

Exemple

Dans cet exemple, les événements sont désactivés avant modification des éléments de la collection ; ainsi l'objet DataSet ne peut pas actualiser les contrôles et avoir une incidence sur les performances.

```
// Désactiver des événements pour l'ensemble de données
monEnsembleDeDonnées.disableEvents();
monEnsembleDeDonnées.last();
while(monEnsembleDeDonnées.hasPrevious()) {
  var prix = monEnsembleDeDonnées.price;
  prix = price * 0.5; // Réduction de 50 % !
  monEnsembleDeDonnées.price = prix;
  monEnsembleDeDonnées.previous();
}
// Indiquer à l'ensemble de données qu'il doit mettre à jour les contrôles
  maintenant.
monEnsembleDeDonnées.enableEvents();
```

Voir aussi

`DataSet.enableEvents()`

DataSet.enableEvents()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

`dataSet``.enableEvents()`

Renvoie

Rien.

Description

Méthode : réactive des événements pour les objets DataSet une fois les événements désactivés par un appel à `DataSet.disableEvents()`. Pour réactiver des événements pour l'objet DataSet, la méthode `enableEvents()` doit être appelée au moins autant de fois que la méthode `disableEvents()` a été appelée.

Exemple

Dans cet exemple, les événements sont désactivés avant modification des éléments de la collection ; ainsi l'objet DataSet ne peut pas actualiser les contrôles et avoir une incidence sur les performances.

```
// Désactiver des événements pour l'ensemble de données
monEnsembleDeDonnées.disableEvents();
monEnsembleDeDonnées.last();
while(monEnsembleDeDonnées.hasPrevious()) {
  var prix = monEnsembleDeDonnées.price;
  prix = price * 0.5; // Réduction de 50 % !
  monEnsembleDeDonnées.price = prix;
  monEnsembleDeDonnées.previous();
}
// Indiquer à l'ensemble de données qu'il doit mettre à jour les contrôles
  maintenant
monEnsembleDeDonnées.enableEvents();
```

Voir aussi

`DataSet.disableEvents()`

DataSet.filtered

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

`dataSet``.filtered`

Description

Propriété : une valeur booléenne indiquant si les données de l'itérateur courant sont filtrées. Lorsqu'elle est définie sur `true`, la fonction de filtre spécifiée par `DataSet.filterFunc` est appelée pour chaque élément de la collection.

Exemple

Dans l'exemple suivant, le filtre est activé sur l'objet DataSet appelé `ed_personnel`. Supposons que chaque enregistrement de la collection possède un champ nommé `typePerso`. La fonction de filtre suivante renvoie `true` si le champ `typePerso` dans l'élément courant est défini sur `"gestion"` ; sinon, elle renvoie `false`.

```
ed_personnel.filtered = true;
ed_personnel.filterFunc = function(item:Object) {
  // Filtrer le personnel au poste de responsable...
  return(item.empType != "gestion");
}
```

Voir aussi

`DataSet.filterFunc`

DataSet.filterFunc

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
dataSet.filterFunc = function(item:Object) {
  // renvoyer true|false;
};
```

Description

Propriété : spécifie une fonction qui détermine les éléments inclus dans l'affichage courant de la collection. Lorsque `DataSet.filtered` est définie sur `true`, la fonction affectée à cette propriété est appelée pour chaque objet de transfert de la collection. Pour chaque élément transmis à la fonction, elle doit renvoyer `true` si l'élément doit être inclus dans l'affichage courant ou `false` dans le cas contraire.

Exemple

Dans l'exemple suivant, le filtre est activé sur l'objet DataSet appelé `ed_personnel`. La fonction de filtre spécifiée renvoie `true` si le champ `typePerso` dans chaque élément est défini sur `"gestion"` ; sinon, elle renvoie `false`.

```
ed_personnel.filtered = true;
ed_personnel.filterFunc = function(item:Object) {
  // Filtrer le personnel au poste de responsable...
  return(item.empType != "gestion");
}
```

Voir aussi

`DataSet.filtered`

DataSet.find()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
dataSet.find(valeursRecherche)
```

Paramètres

`valeursRecherche` Un tableau contenant une ou plusieurs valeurs de champ pouvant être localisées dans le tri courant.

Renvoie

Renvoie `true` si les valeurs sont trouvées ; sinon, renvoie `false`.

Description

Méthode : cherche dans l'affichage courant de la collection un élément avec des valeurs de champ spécifiées par *valeursRecherche*. Les éléments qui se trouvent dans l'affichage courant dépendent des paramètres de filtre et d'étendue courants. Lorsqu'un élément est trouvé, celui-ci devient l'élément courant dans l'objet DataSet.

Les valeurs spécifiées par *valeursRecherche* doivent être dans le même ordre que la liste de champs spécifiée par le tri courant (consultez l'exemple ci-dessous).

Si le tri courant n'est pas unique, l'objet de transfert trouvé est non déterminant. Si vous souhaitez trouver la première ou la dernière occurrence d'un objet de transfert dans un tri qui n'est pas unique, utilisez `DataSet.findFirst()` ou `DataSet.findLast()`.

La conversion des données spécifiées est basée sur le type du champ sous-jacent, ainsi que sur celui spécifié dans le tableau. Par exemple, si vous spécifiez `["05-02-02"]` comme valeur de recherche, le champ de date sous-jacent est utilisé pour convertir la valeur en utilisant la méthode `DataType.setAsString()` de la date. Si vous spécifiez `[new Date().getTime()]`, la méthode `DataType.setAsNumber()` de la date est utilisée.

Exemple

Cet exemple recherche un élément dans la collection courante dont les champs `nom` et `id` contiennent respectivement les valeurs `"Paul"` et 105. Si l'élément est trouvé, la méthode `DataSet.getItemId()` est utilisée pour obtenir l'identifiant unique de l'élément dans la collection et la méthode `DataSet.locateById()` permet de positionner l'itérateur courant sur cet élément.

```
var idEtudiant:String = null;
donnéesEtudiant.addSort("id", ["nom","id"]);
// Localiser l'objet de transfert identifié par "Paul" et 105.
// Notez que l'ordre des champs de recherche correspond aux champs
// spécifiés dans la méthode addSort().
if(donnéesEtudiant.find(["Paul", 105])) {
  idEtudiant = donnéesEtudiant.getItemId();
}
// A présent, utilisez la méthode locateByID() pour positionner l'itérateur
// courant sur l'élément de la collection dont l'ID correspond à idEtudiant
if(idEtudiant != null) {
  donnéesEtudiant.locateById(idEtudiant);
}
```

Voir aussi

`DataSet.applyUpdates()`, `DataSet.getItemId()`, `DataSet.locateById()`

DataSet.findFirst()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

dataSet.findFirst(*valeursRecherche*)

Paramètres

valeursRecherche Un tableau contenant une ou plusieurs valeurs de champ pouvant être localisées dans le tri courant.

Renvoie

Renvoie `true` si les éléments sont trouvés ; sinon, renvoie `false`.

Description

Méthode : recherche dans l'affichage courant de la collection le premier élément avec les valeurs de champ spécifiées par *valeursRecherche*. Les éléments qui se trouvent dans l'affichage courant dépendent des paramètres de filtre et d'étendue courants.

Les valeurs spécifiées par *valeursRecherche* doivent être dans le même ordre que la liste de champs spécifiée par le tri courant (consultez l'exemple ci-dessous).

La conversion des données spécifiées est basée sur le type du champ sous-jacent, ainsi que sur celui spécifié dans le tableau. Par exemple, si la valeur de recherche spécifiée est `["05-02-02"]`, le champ de date sous-jacent est utilisé pour convertir la valeur à l'aide de la méthode setAsString() de la date. Si la valeur spécifiée est `[new Date().getTime()]`, la méthode `setAsNumber()` de la date est utilisée.

Exemple

Cet exemple recherche le premier élément de la collection courante dont les champs `nom` et âge contiennent `"Paul"` et `"13"`. Si l'élément est trouvé, `DataSet.getItemId()` est utilisée pour obtenir l'identifiant unique de l'élément dans la collection et `DataSet.locateById()` permet de positionner l'itérateur courant sur cet élément.

```
var idEtudiant:String = null;
donnéesEtudiant.addSort("nameAndAge", ["nom", "âge");
// Localiser le premier objet de transfert ayant les valeurs spécifiées.
// Notez que l'ordre des champs de recherche correspond aux champs
// spécifiés dans la méthode addSort().
if(donnéesEtudiant.findFirst(["Paul", "13"])) {
  idEtudiant = donnéesEtudiant.getItemId();
}
// A présent, utilisez la méthode locateByID() pour positionner l'itérateur
// courant sur l'élément de la collection dont l'ID correspond à idEtudiant
if(idEtudiant != null) {
  donnéesEtudiant.locateById(idEtudiant);
}
```

Voir aussi

`DataSet.applyUpdates()`, `DataSet.getItemId()`, `DataSet.locateById()`

DataSet.findLast()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

dataSet.findLast(*valeursRecherche*)

Paramètres

valeursRecherche Un tableau contenant une ou plusieurs valeurs de champ pouvant être localisées dans le tri courant.

Renvoie

Renvoie `true` si les éléments sont trouvés ; sinon, renvoie `false`.

Description

Méthode : recherche dans l'affichage courant de la collection le dernier élément ayant les valeurs de champ spécifiées par *valeursRecherche*. Les éléments qui se trouvent dans l'affichage courant dépendent des paramètres de filtre et d'étendue courants.

Les valeurs spécifiées par *valeursRecherche* doivent être dans le même ordre que la liste de champs spécifiée par le tri courant (consultez l'exemple ci-dessous).

La conversion des données spécifiées est basée sur le type du champ sous-jacent, ainsi que sur celui spécifié dans le tableau. Par exemple, si la valeur de recherche spécifiée est `["05-02-02"]`, le champ de date sous-jacent est utilisé pour convertir la valeur à l'aide de la méthode `setAsString()` de la date. Si la valeur spécifiée est `[new Date().getTime()]`, la méthode `setAsNumber()` de la date est utilisée.

Exemple

Cet exemple recherche le dernier élément de la collection courante dont les champs `nom` et `âge` contiennent `"Paul"` et `"13"`. Si l'élément est trouvé, la méthode `DataSet.getItemId()` est utilisée pour obtenir l'identifiant unique de l'élément dans la collection et la méthode `DataSet.locateById()` permet de positionner l'itérateur courant sur cet élément.

```
var idEtudiant:String = null;
donnéesEtudiant.addSort("nameAndAge", ["nom", "âge");
// Localiser le dernier objet de transfert ayant les valeurs spécifiées.
// Notez que l'ordre des champs de recherche correspond aux champs
// spécifiés dans la méthode addSort().
if(donnéesEtudiant.findLast(["Paul", "13"])) {
  idEtudiant = donnéesEtudiant.getItemId();
}
// A présent, utilisez la méthode locateByID() pour positionner l'itérateur
// itérateur sur l'élément de la collection dont l'ID correspond à idEtudiant.
if(idEtudiant != null) {
  donnéesEtudiant.locateById(idEtudiant);
}
```

Voir aussi

DataSet.applyUpdates(), DataSet.getItemId(), DataSet.locateById()

DataSet.first()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

dataSet.first()

Renvoie

Rien.

Description

Méthode : transforme le premier élément de l'affichage courant de la collection en élément courant. Les éléments qui se trouvent dans l'affichage courant dépendent des paramètres de filtre et d'étendue courants.

Exemple

Le code suivant positionne les `donnéesUtilisateur` DataSet au premier élément de sa collection, puis affiche la valeur de la propriété `price` que contient cet élément en utilisant la propriété DataSet.currentItem.

```
inventoryData.first();
trace("Le prix du premier élément est :" + inventoryData.currentItem.price);
```

Voir aussi

DataSet.last()

DataSet.getItemId()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

dataSet.getItemId([*index*])

Paramètres

index Un nombre spécifiant l'élément de l'affichage courant d'éléments dont l'ID est recherché. Ce paramètre est facultatif.

Renvoie

Une chaîne.

Description

Méthode : renvoie l'identifiant de l'élément courant de la collection ou celui de l'élément spécifié par l'*index*. Cet identifiant est unique uniquement dans cette collection et est affecté automatiquement par `DataSet.addItem()`.

Exemple

Le code suivant obtient l'ID unique pour l'élément courant de la collection et l'affiche dans le panneau de sortie.

```
var numElément:String = monEnsembleDeDonnées.getItemId();
trace("id Personnel("+ numElément+ ")");
```

Voir aussi

`DataSet.addItem()`

DataSet.getIterator()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
dataSet.getIterator()
```

Renvoie

Un objet ValueListIterator.

Description

Méthode : renvoie un nouvel itérateur pour cette collection. Cet itérateur est le clone de l'itérateur courant, notamment de sa position courante dans la collection. Cette méthode s'adresse principalement aux utilisateurs expérimentés souhaitant avoir accès à plusieurs affichages simultanés de la même collection.

Exemple

```
monItérateur:ValueListIterator = monEnsembleDeDonnées.getIterator();
monItérateur.sortOn(["name"]);
monItérateur.find({name:"Jean Simon"}).phone = "555-1212";
```

DataSet.hasNext()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
dataSet.hasNext()
```

Renvoie

Une valeur booléenne.

Description

Méthode : renvoie `false` si l'itérateur courant est à la fin de l'affichage de la collection ; sinon, renvoie `true`.

Exemple

Cet exemple itère sur tous les éléments de l'affichage courant de la collection (en commençant par le début), puis effectue un calcul relatif à la propriété `price` de chaque élément.

```
monEnsembleDeDonnées.first();
while (monEnsembleDeDonnées.hasNext() ) {
  var prix = monEnsembleDeDonnées.currentItem.price;
  prix = price * 0.5; // Réduction de 50 % !
  monEnsembleDeDonnées.currentItem.price = prix;
monEnsembleDeDonnées.next();
}
```

Voir aussi

`DataSet.currentItem`, `DataSet.first()`, `DataSet.next()`

DataSet.hasPrevious()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

`dataSet``.hasPrevious()`

Renvoie

Une valeur booléenne.

Description

Méthode : renvoie `false` si l'itérateur courant est au début de l'affichage de la collection ; sinon, renvoie `true`.

Exemple

Cet exemple itère sur tous les éléments de l'affichage courant de la collection (en commençant par le dernier élément), puis effectue un calcul relatif à la propriété `price` de chaque élément.

```
monEnsembleDeDonnées.last();
while(monEnsembleDeDonnées.hasPrevious()) {
  var prix = monEnsembleDeDonnées.currentItem.price;
  prix = price * 0.5; // Réduction de 50 % !
  monEnsembleDeDonnées.currentItem.price = prix;
  monEnsembleDeDonnées.previous();
}
```

Voir aussi

DataSet.currentItem, DataSet.skip(), DataSet.previous()

DataSet.hasSort()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

dataSet.hasSort(*nomTri*)

Paramètres

nomTri Une chaîne qui contient le nom d'un tri créé avec DataSet.addSort()

Renvoie

Une valeur booléenne.

Description

Méthode : renvoie `true` si le tri spécifié par *nomTri* existe ; sinon, renvoie `false`.

Exemple

Le code suivant vérifie qu'un tri nommé « triClient » existe. Si le tri existe déjà, il devient le tri courant via la méthode DataSet.useSort(). Si un tri de ce nom n'existe pas, il est créé au moyen de la méthode DataSet.addSort().

```
if(monEnsembleDeDonnées.hasSort("triClient"))
  monEnsembleDeDonnées.useSort("triClient");
} else {
  monEnsembleDeDonnées.addSort("triClient", ["client"],
  DataSetIterator.Descending);
}
```

Voir aussi

DataSet.applyUpdates(), DataSet.useSort()

DataSet.isEmpty()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

dataSet.isEmpty()

Renvoie

Une valeur booléenne.

Description

Méthode : renvoie `true` si l'objet DataSet spécifié ne contient aucun objet (si *`dataSet`*`.length == 0`).

Exemple

L'exemple suivant désactive un bouton Supprimer l'enregistrement (masqué) si l'objet DataSet auquel il s'applique est vide.

```
if(donnéesUtilisateur.isEmpty()){
    delete_btn.enabled = false;
}
```

Voir aussi

`DataSet.length`

DataSet.items

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

`monEnsembleDeDonnées``.items`

Description

Propriété : un tableau d'éléments gérés par *`monEnsembleDeDonnées`*.

Exemple

Cet exemple affecte un tableau d'objets à une propriété `items` de l'objet DataSet.

```
var recData = [{id:0, firstName:"Michel", lastName:"Jourdain"},
               {id:1, firstName:"Jean", lastName:"Sadourni"},
               {id:2, firstName:"Paul", lastName:"Simon"}];
monEnsembleDeDonnées.items = recData;
```

DataSet.itemClassName

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

`dataSet``.itemClassName`

Description

Propriété : une chaîne indiquant le nom de la classe qui doit être créée lorsque des éléments sont ajoutés à la collection. La classe que vous spécifiez doit implémenter l'interface TransferObject, comme indiqué ci-dessous.

```
interface mx.data.to.TransferObject {
  function clone():Object;
  function getPropertyData():Object;
  function setPropertyData(propData:Object):Void;
}
```

Vous pouvez également définir cette propriété dans l'inspecteur des propriétés.

Pour que cette classe soit disponible à l'exécution, vous devez également inclure une référence pleinement qualifiée à cette classe au sein du code de votre fichier SWF, comme dans le code suivant :

```
var monElément:my.package.monElément;
```

Une exception `DataSetError` est émise si vous essayez de modifier la valeur de cette propriété après le chargement du tableau `DataSet.items`.

Pour plus d'informations sur l'interface TransferObject, consultez *Interface TransferObject*, page 556.

DataSet.iteratorScrolled

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
on(iteratorScrolled) {
  // insérez votre code ici
}
objetDécoute = new Object();
objetDécoute.iteratorScrolled = function (objEvt) {
  // insérez votre code ici
}
dataSet.addEventListener("iteratorScrolled", objetDécoute)
```

Description

Evénement : généré immédiatement lorsque l'itérateur courant a atteint un nouvel élément de la collection.

L'objet événement (*objEvt*) possède les propriétés suivantes :

`target` L'objet DataSet qui a généré l'événement.

`type` La chaîne `"iteratorScrolled"`.

`scrolled` Un nombre qui spécifie le nombre d'éléments que l'itérateur a fait défiler ; les valeurs positives indiquent que l'itérateur a défilé vers le bas dans la collection et les valeurs négatives que l'itérateur a défilé vers le haut dans la collection.

Exemple

Dans cet exemple, la barre d'état d'une application (masquée) est mise à jour lorsque la position de l'itérateur courant change.

```
on(iteratorScrolled) {
  var dataSet:mx.data.components.DataSet = eventObj.target;
  var texteBarreEtat = dataSet.fullname+" Acct #:
  "+dataSet.getField("acctnum").getAsString();
  setStatusBar(texteBarreEtat);
}
```

DataSet.last()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
dataSet.last()
```

Renvoie

Rien.

Description

Méthode : transforme le dernier élément de l'affichage courant de la collection en élément courant.

Exemple

Le code suivant, associé à un composant Button, permet d'atteindre le dernier élément de la collection DataSet.

```
function goLast(objEvt:obj) {
  inventoryData.last();
}
goLast_btn.addEventListener("click", goLast);
```

Voir aussi

DataSet.first()

DataSet.length

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
dataSet.length
```

Description

Propriété (lecture seule) : spécifie le nombre d'éléments dans l'affichage courant de la collection. Le nombre d'éléments pouvant être affichés est basé sur les paramètres de filtre et d'étendue courants.

Exemple

L'exemple suivant prévient les utilisateurs lorsque ceux-ci n'ont pas intégré suffisamment d'entrées dans l'ensemble de données, probablement en utilisant un composant DataGrid modifiable.

```
if(monEnsembleDeDonnées.length < MIN_REQUIRED) {
  alert("Vous avez besoin d'au moins "+MIN_REQUIRED);
}
```

DataSet.loadFromSharedObj()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

dataSet.loadFromSharedObj(*nomObjet*, [*cheminLocal*])

Paramètres

nomObjet Une chaîne spécifiant le nom de l'objet partagé à récupérer. Le nom peut inclure des barres obliques (par exemple, « travail/adresses »). Les espaces et les caractères suivants ne sont pas autorisés dans le nom spécifié :

~ % & \ ; : " ' , < > ? #

cheminLocal Un paramètre de chaîne facultatif qui précise le chemin complet ou partiel vers le fichier SWF qui a créé l'objet partagé. La chaîne est utilisée pour déterminer où l'objet est stocké sur l'ordinateur de l'utilisateur. La valeur par défaut est le chemin complet du fichier SWF.

Renvoie

Rien.

Description

Méthode : charge toutes les données pertinentes nécessaires à la restauration de cette collection DataSet à partir d'un objet partagé. Pour enregistrer une collection DataSet dans un objet partagé, utilisez DataSet.saveToSharedObj(). La méthode DataSet.loadFromSharedObject() remplace toute donnée ou modification en attente susceptible d'être présente dans cette collection DataSet. Notez que le nom d'occurrence de la collection DataSet est utilisé pour identifier les données dans l'objet partagé spécifié.

Cette méthode émet une exception DataSetError si l'objet partagé spécifié ne peut pas être trouvé ou si un problème survient lors de la récupération des données qu'il contient.

Exemple

Cet exemple tente de charger un objet partagé appelé `webapp/customerInfo` associé à l'ensemble de données `monEnsembleDeDonnées`. La méthode est appelée au sein d'un bloc de code `try...catch`.

```
try
  monEnsembleDeDonnées.loadFromSharedObj("webapp/customerInfo");
}
catch(e:DataSetError) {
  trace("Impossible de charger l'objet partagé.");
}
```

Voir aussi

`DataSet.saveToSharedObj()`

DataSet.locateById()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

`dataSet``.locateById(`*`id`*`)`

Paramètres

`id` Un identifiant de chaîne pour l'élément de la collection devant être localisé.

Renvoie

Une valeur booléenne.

Description

Méthode : positionne l'itérateur courant sur l'élément de la collection dont l'ID correspond à *`id`*. Cette méthode renvoie `true` si l'ID spécifié correspond à celui d'un élément de la collection ; sinon, elle renvoie `false`.

Exemple

Cet exemple utilise `DataSet.find()` pour rechercher un élément dans la collection courante dont les champs `nom` et `id` contiennent respectivement les valeurs `"Paul"` et 105. Si l'élément est trouvé, la méthode `DataSet.getItemId()` est utilisée pour obtenir l'identifiant unique de cet élément et la méthode `DataSet.locateById()` permet de positionner l'itérateur courant sur cet élément.

```
var idEtudiant:String = null;
donnéesEtudiant.addSort("id", ["nom","id"]);
if(donnéesEtudiant.find(["Paul", 105])) {
  idEtudiant = donnéesEtudiant.getItemId();
  donnéesEtudiant.locateById(idEtudiant);
}
```

Voir aussi

DataSet.applyUpdates(), DataSet.find(), DataSet.getItemId()

DataSet.logChanges

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

dataSet.logChanges

Description

Propriété : une valeur booléenne qui spécifie si les modifications apportées à l'ensemble de données ou à ses éléments doivent être enregistrées (true) ou non (false) dans DataSet.deltaPacket.

Lorsque cette propriété est définie sur true, les opérations effectuées au niveau de la collection et au niveau de l'élément sont enregistrées. Les modifications apportées au niveau de la collection comprennent l'ajout et la suppression d'éléments de la collection. Les modifications apportées au niveau de l'élément comprennent les modifications de propriété apportées aux éléments et les appels de méthode effectués sur les éléments via le composant DataSet.

Exemple

L'exemple suivant désactive l'enregistrement pour l'objet DataSet appelé donnéesUtilisateur.

```
donnéesUtilisateur.logChanges = false;
```

Voir aussi

DataSet.deltaPacket

DataSet.modelChanged

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Description

```
on(modelChanged) {
  // insérez votre code ici
}
objetDécoute = new Object();
objetDécoute.modelChanged = function (objEvt) {
  // insérez votre code ici
}
dataSet.addEventListener("modelChanged", objetDécoute)
```

Description

Evénement : diffusé lorsque la collection est modifiée de quelque manière que ce soit (par exemple, lorsque des éléments sont supprimés de la collection ou ajoutés à cette dernière, lorsque la valeur d'une propriété d'élément est modifiée ou lorsque la collection est filtrée ou triée).

L'objet événement (*objEvt*) possède les propriétés suivantes :

`target` L'objet DataSet qui a généré l'événement.

`type` La chaîne `"iteratorScrolled"`.

`premierElément` L'index (numéro) du premier élément de la collection affecté par la modification.

`dernierElément` L'index (numéro) du dernier élément de la collection affecté par la modification (équivaut à `premierElément` si un seul élément a été affecté).

`nomDeChamp` Une chaîne qui contient le nom du champ affecté. Cette propriété est `undefined` sauf si la modification concernait une propriété de l'objet DataSet.

`nomEvénement` Une chaîne qui décrit la modification qui a eu lieu. Cette valeur peut être :

Valeur de chaîne	Description
`"addItems"`	Une série d'éléments a été ajoutée.
`"filterModel"`	Le modèle a été filtré et l'affichage doit être actualisé (réinitialisez la position de défilement).
`"removeItems"`	Une série d'éléments a été supprimée.
`"schemaLoaded"`	La définition des champs du fournisseur de données a été déclarée.
`"sort"`	Les données ont été triées.
`"updateAll"`	L'ensemble de l'affichage doit être actualisé, en excluant la position de défilement.
`"updateColumn"`	Une définition de champ complète dans le fournisseur de données doit être actualisée.
`"updateField"`	Un champ dans un élément a été modifié et doit être actualisé.
`"updateItems"`	Une série d'éléments doit être actualisée.

Exemple

Dans cet exemple, un bouton Supprimer l'élément est désactivé si les éléments ont été supprimés de la collection et que l'objet DataSet cible ne contient plus d'éléments.

```
on(modelChanged) {
  supprimer_btn.enabled = ((objEvt.nomEvénement == "removeItems") &&
  (eventObj.target.isEmpty()));
}
```

Voir aussi

`DataSet.isEmpty()`

DataSet.newItem

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
on(newItem) {
  // insérez votre code ici
}
objetDécoute = new Object();
objetDécoute.newItem = function (objEvt) {
  // insérez votre code ici
}
dataSet.addEventListener("newItem", objetDécoute)
```

Description

Evénement : diffusé lorsqu'un nouvel objet de transfert est construit au moyen de `DataSet.createItem()`. Un écouteur pour cet événement peut modifier l'élément avant son ajout à la collection.

L'objet événement (*objEvt*) possède les propriétés suivantes :

`target` L'objet DataSet qui a généré l'événement.

`type` La chaîne `"iteratorScrolled"`.

`élément` Une référence à l'élément créé.

Exemple

Cet exemple modifie un élément nouvellement créé avant son ajout à la collection.

```
function newItemEvent(evt:Object):Void {
  var personnel:Object = evt.item;
  personnel.name = "nouveauType";
  // les données de propriété sont au format XML
  personnel.zip =
  personnel.getPropertyData().firstChild.childNodes[1].attributes.zip;
}
ed_personnel.addEventListener("newItem", newItemEvent);
```

DataSet.next()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
dataSet.next()
```

Renvoie

Rien.

Description

Méthode : transforme l'élément suivant de l'affichage courant de la collection en élément courant. Les éléments qui se trouvent dans l'affichage courant dépendent des paramètres de filtre et d'étendue courants.

Exemple

Cet exemple boucle sur tous les éléments d'un objet DataSet, en commençant par le premier élément, et effectue un calcul sur un champ dans chaque élément.

```
monEnsembleDeDonnées.first();
while (monEnsembleDeDonnées.hasNext() ) {
  var prix = monEnsembleDeDonnées.price;
  prix = price * 0.5; // Réduction de 50 % !
  monEnsembleDeDonnées.price = prix;
monEnsembleDeDonnées.next();
}
```

Voir aussi

DataSet.first(), DataSet.hasNext()

DataSet.previous()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
dataSet.previous()
```

Renvoie

Rien.

Description

Méthode : transforme l'élément précédent de l'affichage courant de la collection en élément courant. Les éléments qui se trouvent dans l'affichage courant dépendent des paramètres de filtre et d'étendue courants.

Cet exemple boucle sur tous les éléments de l'affichage courant de la collection, en commençant par le dernier élément, et effectue un calcul sur un champ dans chaque élément.

```
monEnsembleDeDonnées.last();
while(monEnsembleDeDonnées.hasPrevious()) {
  var prix = monEnsembleDeDonnées.price;
  prix = price * 0.5; // Réduction de 50 % !
  monEnsembleDeDonnées.price = prix;
  monEnsembleDeDonnées.previous();
}
```

Voir aussi

`DataSet.first()`, `DataSet.hasNext()`

DataSet.properties

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

`dataSet``.properties`

Description

Propriété (lecture seule) : renvoie un objet qui contient toutes les propriétés (champs) exposées pour tout objet de transfert dans cette collection.

Exemple

Cet exemple affiche tous les noms des propriétés dans l'objet DataSet appelé `monEnsembleDeDonnées`.

```
for(var i in monEnsembleDeDonnées.properties) {
  trace("champ '"+i+ "' possède la valeur "+
  monEnsembleDeDonnées.properties[i]);
}
```

DataSet.readOnly

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

`dataSet``.readOnly`

Description

Propriété : une valeur booléenne spécifiant si cette collection peut être modifiée (`false`) ou si elle est en lecture seule (`true`). La définition de cette propriété sur `true` empêche les mises à jour de la collection.

Vous pouvez également définir cette propriété dans l'inspecteur des propriétés.

Exemple

L'exemple suivant transforme l'objet DataSet appelé `monEnsembleDeDonnées` en lecture seule, puis tente de modifier la valeur d'une propriété appartenant à l'élément courant dans la collection. Ceci générera une exception.

```
monEnsembleDeDonnées.readOnly = true;
// Ceci générera une exception
monEnsembleDeDonnées.currentItem.price = 15;
```

Voir aussi

DataSet.currentItem

DataSet.removeAll()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
dataSet.removeAll()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : supprime tous les éléments de la collection DataSet.

Exemple

Cet exemple supprime tous les éléments de la collection DataSet `ed_contact` :

```
ed_contact.removeAll();
```

DataSet.removeItem

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
on(removeItem) {
  // insérez votre code ici
}
objetDécoute = new Object();
objetDécoute.removeItem = function (objEvt) {
  // insérez votre code ici
}
dataSet.addEventListener("removeItem", objetDécoute)
```

Description

Evénement : généré juste avant la suppression d'un nouvel élément de cette collection.

Si vous définissez la propriété `result` de l'objet événement sur `false`, la suppression est annulée ; si vous la définissez sur `true`, elle est autorisée.

L'objet événement (*`objEvt`*) possède les propriétés suivantes :

`target` L'objet DataSet qui a généré l'événement.

`type` La chaîne `"removeItem"`.

`item` Une référence à l'élément de la collection à supprimer.

`result` Une valeur booléenne indiquant si l'élément doit être supprimé. Par défaut, cette valeur est `true`.

Exemple

Dans cet exemple, un gestionnaire d'événement `on(removeItem)` annule la suppression du nouvel élément si une fonction définie par l'utilisateur, appelée `userHasAdminPrivs()`, renvoie `false` ; sinon, la suppression est autorisée.

```
on(removeItem) {
  if(globalObj.userHasAdminPrivs()) {
    // Autoriser la suppression de l'élément.
    objEvt.result = true;
} else {
    // Ne pas autoriser la suppression de l'élément ; l'utilisateur ne dispose
  pas des droits d'administrateur.
    eventObj.result = false;
  }
}
```

Voir aussi

`DataSet.addItem`

DataSet.removeItem()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

`dataSet``.removeItem([`*`item`*`])`

Paramètres

`item` L'élément à supprimer. Ce paramètre est facultatif.

Renvoie

Une valeur booléenne. Renvoie `true` si l'élément a été supprimé avec succès ; sinon, renvoie `false`.

Description

Méthode : supprime l'élément spécifié de la collection ou supprime l'élément courant si le paramètre *item* est omis. Cette opération est enregistrée dans DataSet.deltaPacket si DataSet.logChanges est true.

Exemple

Le code suivant, associé à une occurrence du composant Button, supprime l'élément courant de l'objet DataSet appelé donnéesUtilisateur résidant sur le même scénario que l'occurrence Button.

```
on(click) {
  _parent.donnéesUtilisateur.removeItem();
}
```

Voir aussi

DataSet.deltaPacket, DataSet.logChanges

DataSet.removeRange()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
dataSet.removeRange()
```

Renvoie

Rien.

Description

Méthode : supprime les paramètres de point de fin spécifiés au moyen de DataSet.setRange() pour l'itérateur courant.

Exemple

```
monEnsembleDeDonnées.addSort("name_id", ["nom", "id"]);
monEnsembleDeDonnées.setRange(["Paul", 105],["Catherine", 110]);
while (monEnsembleDeDonnées.hasNext() ) {
  monEnsembleDeDonnées.gradeLevel ="5"; // modifier tous les chiffres dans
  cette étendue
monEnsembleDeDonnées.next();
}
monEnsembleDeDonnées.removeRange();
monEnsembleDeDonnées.removeSort("name_id");
```

Voir aussi

DataSet.applyUpdates(), DataSet.hasNext(), DataSet.next(), DataSet.removeSort(), DataSet.setRange()

DataSet.removeSort()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

dataSet.removeSort(*nomTri*)

Paramètres

nomTri Une chaîne spécifiant le nom du tri à supprimer.

Renvoie

Rien.

Description

Méthode : supprime le tri spécifié de cet objet DataSet, si le tri existe. Si le tri spécifié n'existe pas, cette méthode émet une exception `DataSetError`.

Exemple

```
monEnsembleDeDonnées.addSort("name_id", ["nom", "id"]);
monEnsembleDeDonnées.setRange(["Paul", 105],["Catherine", 110]);
while (monEnsembleDeDonnées.hasNext() ) {
  monEnsembleDeDonnées.gradeLevel ="5"; // modifier tous les chiffres dans
  cette étendue
monEnsembleDeDonnées.next();
}
monEnsembleDeDonnées.removeRange();
monEnsembleDeDonnées.removeSort("name_id");
```

Voir aussi

DataSet.applyUpdates(), DataSet.hasNext(), DataSet.next(), DataSet.removeRange(), DataSet.setRange()

DataSet.resolveDelta

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
on(resolveDelta) {
  // insérez votre code ici
}
objetDécoute = new Object();
objetDécoute.resolveDelta = function (objEvt) {
  // insérez votre code ici
}
dataSet.addEventListener("resolveDelta", objetDécoute)
```

Description

Evénement : diffusé lorsqu'un objet DeltaPacket est affecté à `DataSet.deltaPacket` dont l'ID de transaction correspond à celui d'un objet DeltaPacket récupéré précédemment à partir de l'objet DataSet, et qui possède des messages associés à n'importe quel objet Delta ou DeltaItem dans cet objet DeltaPacket.

Cet événement vous donne la possibilité de reconstituer toute erreur renvoyée par le serveur lors de la tentative d'application des modifications précédemment soumises. En général, vous utilisez cet événement pour afficher une « boîte de dialogue de reconstitution » avec les valeurs en conflit, autorisant l'utilisateur à apporter les modifications appropriées aux données de sorte qu'elles puissent à nouveau être envoyées.

L'objet événement (*objEvt*) possède les propriétés suivantes :

`target` L'objet DataSet qui a généré l'événement.

`type` La chaîne `"resolveDelta"`.

`données` Un tableau d'objets Delta et DeltaItem associés qui possèdent des messages dont la longueur est différente de zéro.

Exemple

Cet exemple affiche un formulaire appelé `formReconstit` (masqué) et appelle une méthode sur cet objet formulaire (`setReconcileData()`) qui autorise l'utilisateur à reconstituer toute valeur en conflit renvoyée par le serveur :

```
monEnsembleDeDonnées.addEventListener("resolveDelta", resolveDelta);
function resolveDelta(objEvt:Object) {
  formReconstit.visible = true;
  formReconstit.setReconcileData(objEvt.data);
}
// dans le code formReconstit
function setReconcileData(data:Array):Void {
  var di:DeltaItem;
  var ops:Array = ["propriété", "méthode"];
  var cl:Array;
  // modifier la liste
  var msg:String;
```

```
    for (var i = 0; i<data.length; i++) {
      cl = data[i].getChangeList();
      for (var j = 0; j<cl.length; j++) {
        di = cl[j];
        msg = di.getMessage();
        if (msg.length>0) {
          trace("Le problème suivant est survenu '"+msg+"' lors de l'opération
    de '"+ops[di.kind]+"' modification sur/avec '"+di.name+"' valeur du serveur
    courant ["+di.curValue+"], valeur envoyée ["+di.newValue+"] Veuillez
    résoudre ce problème !");
        }
      }
    }
  }
```

DataSet.saveToSharedObj()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

dataSet.saveToSharedObj(*nomObjet*, [*cheminLocal*])

Paramètres

nomObjet Une chaîne spécifiant le nom de l'objet partagé à créer. Le nom peut inclure des barres obliques (par exemple, « travail/adresses »). Les espaces et les caractères suivants ne sont pas autorisés dans le nom spécifié :

~ % & \ ; : " ' , < > ? #

cheminLocal Un paramètre de chaîne facultatif qui précise le chemin complet ou partiel vers le fichier SWF qui a créé l'objet partagé. Cette chaîne permet de déterminer l'emplacement où l'objet sera stocké sur l'ordinateur de l'utilisateur. La valeur par défaut est le chemin complet du fichier SWF.

Renvoie

Rien.

Description

Méthode : enregistre toutes les données pertinentes nécessaires à la restauration de cette collection DataSet dans un objet partagé. Ceci permet aux utilisateurs de travailler lorsqu'ils sont déconnectés des données source, s'il s'agit d'une ressource réseau. Cette méthode remplace toute donnée susceptible d'être présente dans l'objet partagé spécifié pour cette collection DataSet. Pour restaurer une collection DataSet à partir d'un objet partagé, utilisez DataSet.loadFromSharedObj(). Notez que le nom d'occurrence de la collection DataSet est utilisé pour identifier les données dans l'objet partagé spécifié.

Si l'objet partagé ne peut pas être créé ou qu'un problème survient lors de la purge des données vers cet objet, cette méthode émet une exception DataSetError.

Exemple

Cet exemple appelle `saveToSharedObj()` dans un bloc `try..catch` et affiche un message d'erreur si un problème survient lors de l'enregistrement des données dans l'objet partagé.

```
try
  monEnsembleDeDonnées.saveToSharedObj("webapp/customerInfo");
}
catch(e:DataSetError) {
  trace("Impossible de créer l'objet partagé");
}
```

Voir aussi

`DataSet.loadFromSharedObj()`

DataSet.schema

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

`dataSet``.schema`

Description

Propriété : fournit la représentation XML du schéma pour cet objet DataSet. Le code XML affecté à cette propriété doit avoir le format suivant :

```
<?xml version="1.0" ?>
<propriétés>
  <property name="propertyName">
    <type name="dataType" />
    <encoder name="dataType">
      <options>
        <dataFormat>options de format<dataFormat/>
      <options/>
    <encoder/>
    <kind name="dataKind">
      <options/>
    </kind>
  </property>
  <property> ... </property>
  ...
</properties>
```

Une exception `DataSetError` est émise si le code XML spécifié ne respecte pas le format ci-dessus.

Exemple

```
monEnsembleDeDonnées.schema = new XML("<properties><property
  name="facturable"> ..etc.. </properties>");
```

DataSet.selectedIndex

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
dataSet.selectedIndex
```

Description

Propriété : spécifie l'index sélectionné dans la collection. Vous pouvez lier cette propriété à l'élément sélectionné dans un composant DataGrid ou List, et inversement. Pour obtenir un exemple complet présentant ce point, consultez *Création d'une application avec le composant DataSet*, page 208.

Exemple

L'exemple suivant définit l'index sélectionné d'un objet DataSet (`donnéesUtilisateur`) sur l'index sélectionné dans un composant DataGrid (`grilleUtilisateur`).

```
donnéesUtilisateur.selectedIndex = grilleUtilisateur.selectedIndex;
```

DataSet.setIterator()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
dataSet.setIterator(itérateur)
```

Paramètres

itérateur Un objet itérateur renvoyé par un appel à `DataSet.getIterator()`.

Renvoie

Rien.

Description

Méthode : affecte l'itérateur spécifié à cet objet DataSet et le transforme en itérateur courant. L'itérateur spécifié doit venir d'un appel précédent à `DataSet.getIterator()` sur l'objet DataSet auquel il est affecté ; sinon, une exception `DataSetError` est émise.

Exemple

```
monItérateur:ValueListIterator = monEnsembleDeDonnées.getIterator();
monItérateur.sortOn(["name"]);
monEnsembleDeDonnées.setIterator(monItérateur);
```

Voir aussi

DataSet.getIterator()

DataSet.setRange()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

dataSet.setRange(*valeursDépart*, *valeursFin*)

Paramètres

valeursDépart Un tableau des principales valeurs des propriétés du premier objet de transfert de l'étendue.

valeursFin Un tableau des principales valeurs des propriétés du dernier objet de transfert de l'étendue.

Renvoie

Rien.

Description

Méthode : définit les points de fin pour l'itérateur courant. Les points de fin définissent une étendue au sein de laquelle l'itérateur agit. Ceci est uniquement valable si un tri valide a été défini pour l'itérateur courant au moyen de DataSet.applyUpdates().

Si vous souhaitez obtenir un groupement de valeurs, il est plus efficace de définir une étendue pour l'itérateur courant plutôt que d'utiliser une fonction de filtre (consultez DataSet.filterFunc).

Exemple

```
monEnsembleDeDonnées.addSort("name_id", ["nom", "id"]);
monEnsembleDeDonnées.setRange(["Paul", 105],["Catherine", 110]);
while (monEnsembleDeDonnées.hasNext() ) {
  monEnsembleDeDonnées.gradeLevel ="5"; // modifier tous les chiffres dans
  cette étendue
monEnsembleDeDonnées.next();
}
monEnsembleDeDonnées.removeRange();
monEnsembleDeDonnées.removeSort("name_id");
```

Voir aussi

DataSet.applyUpdates(), DataSet.hasNext(), DataSet.next(), DataSet.removeRange(), DataSet.removeSort()

DataSet.skip()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

dataSet.skip(*décalage*)

Paramètres

décalage Un entier spécifiant le nombre d'enregistrements en fonction duquel déplacer la position de l'itérateur.

Renvoie

Rien.

Description

Méthode : avance ou recule la position de l'itérateur dans la collection selon le nombre spécifié par le *décalage*. Les valeurs de *décalage* positives avancent la position de l'itérateur ; les valeurs négatives la font reculer.

Si le *décalage* spécifié se situe au-delà du début (ou de la fin) de la collection, l'itérateur se place au début (ou à la fin) de la collection.

Exemple

Cet exemple positionne l'itérateur courant au premier élément de la collection, puis se déplace vers l'avant-dernier élément, et effectue un calcul sur un champ appartenant à cet élément.

```
monEnsembleDeDonnées.first();
// Déplacer vers l'élément situé juste avant le dernier
var élémentsAIgnorer = monEnsembleDeDonnées.length - 2;
monEnsembleDeDonnées.skip(élémentsAIgnorer).price =
  monEnsembleDeDonnées.amount * 10;
```

DataSet.useSort()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

dataSet.useSort(*nomTri*, *ordre*)

Paramètres

nomTri Une chaîne qui contient le nom du tri à utiliser.

ordre Une valeur entière qui indique l'ordre de tri ; la valeur doit être `DataSetIterator.Ascending` ou `DataSetIterator.Descending`.

Renvoie

Rien.

Description

Méthode : fait passer le tri pour l'itérateur courant vers le tri spécifié par *nomTri*, si celui existe. Si le tri spécifié par *nomTri* n'existe pas, une exception `DataSetError` est émise.

Pour créer un tri, utilisez `DataSet.applyUpdates()`.

Exemple

Ce code utilise `DataSet.hasSort()` pour déterminer si un tri appelé `"client"` existe. Si c'est le cas, le code appelle `DataSet.useSort()` pour que `"client"` devienne le tri courant. Sinon, le code crée un tri de ce nom en utilisant `DataSet.addSort()`.

```
if(monEnsembleDeDonnées.hasSort("client")) {
  monEnsembleDeDonnées.useSort("client");
} else {
  monEnsembleDeDonnées.addSort("client", ["client"],
  DataSetIterator.Descending);
}
```

Voir aussi

`DataSet.applyUpdates()`, `DataSet.hasSort()`

Composant DateChooser (Flash Professionnel uniquement)

Le composant DateChooser est un calendrier qui permet aux utilisateurs de sélectionner une date. Il comprend un bouton qui permet aux utilisateurs de faire défiler les mois et de sélectionner une date en cliquant sur celle de leur choix. Vous pouvez définir des paramètres qui indiquent les noms des mois et des jours, le premier jour de la semaine et les dates désactivées, et qui permettent la mise en surbrillance de la date courante.

Un aperçu en direct de chaque occurrence DateChooser reflète les valeurs indiquées par le panneau de l'inspecteur des propriétés ou des composants au cours de la programmation.

Utilisation du composant DateChooser (Flash Professionnel uniquement)

Le composant DateChooser peut être utilisé là où vous souhaitez qu'un utilisateur sélectionne une date. Par exemple, vous pouvez utiliser un composant DateChooser dans un système de réservation d'hôtel comprenant des dates sélectionnables et des dates désactivées. Vous pouvez également utiliser le composant DateChooser dans une application qui affiche des événements courants, tels que des spectacles ou des réunions, lorsque l'utilisateur sélectionne une date.

Paramètres DateChooser

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

monthNames définit les noms des mois affichés dans la ligne d'en-tête du calendrier. La valeur est un tableau et la valeur par défaut est ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October","November", "December"].

dayNames définit les noms des jours de la semaine. La valeur est un tableau et la valeur par défaut est ["S", "M", "T", "W", "T", "F", "S"].

firstDayOfWeek indique le jour de la semaine (0-6, 0 étant le premier élément du tableau `dayNames`) affiché dans la première colonne de DateChooser. Cette propriété modifie l'ordre d'affichage des colonnes des jours.

disabledDays indique les jours désactivés de la semaine. Ce paramètre est un tableau qui peut comprendre un maximum de 7 valeurs. La valeur par défaut est [] (tableau vide).

showToday indique si la date du jour doit être mise en surbrillance ou non. La valeur par défaut est `true`.

Vous pouvez rédiger du code ActionScript pour contrôler ces options et d'autres options du composant DateChooser à l'aide des propriétés, méthodes et événements ActionScript. Pour plus d'informations, consultez *Classe DateChooser (Flash Professionnel uniquement)*, page 255.

Création d'une application avec le composant DateChooser

La procédure suivante explique comment ajouter un composant DateChooser à une application lors de la programmation. Dans cet exemple, le composant DateChooser autorise l'utilisateur à choisir une date dans un système de réservation de vols. Toutes les dates précédant le 15 octobre doivent être désactivées. Une plage de dates au mois de décembre doit également être désactivée afin de créer une période correspondant à des vacances ; les lundis doivent également être désactivés.

Pour créer une application avec le composant DateChooser, effectuez les opérations suivantes :

1. Dans le panneau Composants, double-cliquez sur le composant DateChooser afin d'ajouter ce dernier à la scène.
2. Dans l'inspecteur des propriétés, entrez le nom d'occurrence **calendrierVols**.
3. Dans le panneau Actions, entrez le code suivant dans l'image 1 du scénario pour définir la plage des dates sélectionnables :

   ```
   calendrierVols.selectableRange = {rangeStart:new Date(2003, 9, 15),
     rangeEnd:new Date(2003, 11, 31)}
   ```

   Ce code affecte une valeur à la propriété `selectableRange` dans un objet ActionScript qui contient deux objets Date avec les noms de variables `rangeStart` et `rangeEnd`. Il définit les extrémités inférieure et supérieure de la plage dans laquelle l'utilisateur peut choisir une date.
4. Dans le panneau Actions, entrez le code suivant dans l'Image 1 du scénario pour définir une plage de dates désactivées correspondant aux vacances.

   ```
   calendrierVols.disabledRanges = [{rangeStart: new Date(2003, 11, 15),
     rangeEnd: new Date(2003, 11, 26)}];
   ```
5. Dans le panneau Actions, entrez le code suivant dans l'image 1 du scénario pour désactiver les lundis :

   ```
   calendrierVols.disabledDays=[1];
   ```
6. Choisissez Contrôle > Tester l'animation.

Personnalisation du composant DateChooser (Flash Professionnel uniquement)

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

Utilisation de styles avec le composant DateChooser

Vous pouvez définir des propriétés de style pour modifier l'apparence d'une occurrence de sélecteur de dates. 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 29.

Un composant DateChooser prend en charge les styles de halo suivants :

Style	Description
`themeColor`	La couleur de l'éclat pour les dates survolées et sélectionnées. 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"`.

Utilisation des enveloppes avec le composant DateChooser

Le composant DateChooser utilise des enveloppes pour représenter ses états visuels. Pour appliquer une enveloppe au composant DateChooser au cours de la programmation, modifiez les symboles d'enveloppe dans le dossier Flash UI Components 2/Themes/MMDefault/DateChooser Assets/Elements/Month skins states, dans la bibliothèque de l'un des fichiers FLA de thèmes. Pour plus d'informations, consultez *A propos de l'application des enveloppes aux composants*, page 38.

Seuls les boutons de défilement des mois peuvent être enveloppés dynamiquement dans ce composant. Un composant DateChooser utilise les propriétés d'enveloppe suivantes :

Propriété	Description
`falseUpSkin`	Etat Relevé. Les valeurs par défaut sont `fwdMonthUp` et `backMonthUp`.
`falseDownSkin`	L'état Enfoncé. Les valeurs par défaut sont `fwdMonthDown` et `backMonthDown`.
`falseDisabledSkin`	Etat Désactivé. Les valeurs par défaut son `fwdMonthDisabled` et `backMonthDisabled`.

Classe DateChooser (Flash Professionnel uniquement)

Héritage UIObject > UIComponent > DateChooser

Nome de classe ActionScript mx.controls.DateChooser

Les propriétés de la classe DateChooser vous permettent d'accéder à la date sélectionnée et au mois et à l'année affichés. Vous pouvez également définir les noms des jours et des mois, indiquer les dates désactivées et les dates sélectionnables, définir le premier jour de la semaine et indiquer si la date courante doit être mise en surbrillance.

La définition d'une propriété de la classe DateChooser 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.DateChooser.version);
```

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

Méthodes de la classe DateChooser

Hérite de toutes les méthodes des classes *UIObject* et *UIComponent*.

Propriétés de la classe DateChooser

Propriété	Description
`DateChooser.dayNames`	Un tableau indiquant les noms des jours de la semaine.
`DateChooser.disabledDays`	Un tableau indiquant les jours de la semaine qui sont désactivés pour toutes les dates applicables dans le sélecteur de dates.
`DateChooser.disabledRanges`	Une plage de dates désactivée ou une seule date désactivée.
`DateChooser.displayedMonth`	Un nombre indiquant un élément du tableau `monthNames` à afficher dans le sélecteur de dates.
`DateChooser.displayedYear`	Un nombre indiquant l'année à afficher.
`DateChooser.firstDayOfWeek`	Un nombre indiquant un élément du tableau `dayNames` à afficher dans la première colonne du sélecteur de dates.

Propriété	Description
DateChooser.monthNames	Un tableau de chaînes indiquant les noms des mois.
DateChooser.selectableRange	Une seule date sélectionnable ou une plage de dates sélectionnables.
DateChooser.selectedDate	Un objet Date indiquant la date sélectionnée.
DateChooser.showToday	Une valeur booléenne indiquant si la date courante est mise en surbrillance.

Hérite de toutes les propriétés des classes *UIObject* et *UIComponent*.

Evénements de la classe DateChooser

Evénement	Description
DateChooser.change	Diffusé lorsqu'une date est sélectionnée.
DateChooser.scroll	Diffusé lorsque l'utilisateur clique sur les boutons des mois.

Hérite de tous les événements des classes *UIObject* et *UIComponent*.

DateChooser.change

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

Usage 1 :

```
on(change){
   ...
}
```

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.change = function(objetEvt){
   ...
}
occurrenceDateChooser.addEventListener("change", objetDécoute)
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsqu'une date est sélectionnée.

Le premier exemple d'utilisation fait appel à un gestionnaire `on()` et doit être directement associé à une occurrence de composant DateChooser. 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 sélecteur de dates `monDC`, envoie « _level0.monDC » 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 (*`occurrenceDateChooser`*) distribue un événement (ici, `change`) qui est géré par une fonction, également appelée « *gestionnaire* », sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

Exemple

Cet exemple, écrit sur une image du scénario, envoie un message au panneau de sortie lorsqu'un DateChooser nommé `monDC` 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 `objEvt`, pour générer un message. La propriété `target` d'un objet événement est le composant ayant généré l'événement (dans cet exemple, `monDC`). L'utilisateur accède à la propriété `NumericStepper.maximum` à partir de la propriété `target` de l'objet événement. La dernière ligne appelle la méthode `UIEventDispatcher.addEventListener()` depuis `monDC` et lui transmet l'événement `change` et l'objet d'écoute `form` comme paramètres, comme dans l'exemple suivant :

```
form.change = function(objEvt){
    trace("date sélectionnée " +  eventObj.target.selectedDate) ;
}
monDC.addEventListener("change", form);
```

DateChooser.dayNames

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monDC.dayNames
```

Description

Propriété : un tableau contenant les noms des jours de la semaine. Dimanche correspond au premier jour (à l'emplacement d'index 0) et le reste des jours suit, dans l'ordre. La valeur par défaut est `["S", "M", "T", "W", "T", "F", "S"]`.

Exemple

L'exemple suivant change la valeur du cinquième jour de la semaine (jeudi) en la faisant passer de "T" à "R" :

```
monDC.dayNames[4] = "R";
```

DateChooser.disabledDays

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monDC.disabledDays
```

Description

Propriété : un tableau indiquant les jours désactivés de la semaine. Toutes les dates du mois qui correspondent au jour spécifié sont désactivées. Les éléments de ce tableau peuvent avoir des valeurs comprises entre 0 (dimanche) et 6 (samedi). La valeur par défaut est `[]` (tableau vide).

Exemple

L'exemple suivant désactive les dimanches et les samedis de sorte que l'utilisateur puisse uniquement sélectionner les jours ouvrés :

```
monDC.disabledDays = [0, 6];
```

DateChooser.disabledRanges

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

monDC.disabledRanges

Description

Propriété : désactive un seul jour ou une plage de jours. Cette propriété est un tableau d'objets. Chaque objet du tableau doit être soit un objet Date spécifiant un seul jour à désactiver, soit un objet contenant une des propriétés `rangeStart` ou `rangeEnd`, voire les deux, dont les valeurs doivent être un objet Date. Les propriétés `rangeStart` et `rangeEnd` décrivent les limites de la plage de dates. Si l'une ou l'autre des propriétés est omise, l'étendue est illimitée dans le sens correspondant.

La valeur par défaut de `disabledRanges` est undefined.

Lorsque vous définissez des dates pour la propriété `disabledRanges`, spécifiez des dates complètes. Par exemple, `new Date(2003,6,24)` plutôt que `new Date()`. Si vous ne spécifiez pas une date complète, le temps renvoie `00:00:00`.

Exemple

L'exemple suivant définit un tableau comprenant des objets Date `rangeStart` et `rangeEnd` qui désactivent les dates comprises entre le 7 mai et le 7 juin :

```
monDC.disabledRanges = [ {rangeStart: new Date(2003, 4, 7), rangeEnd: new
  Date(2003, 5, 7)}];
```

L'exemple suivant désactive toutes les dates situées après le 7 novembre :

```
monDC.disabledRanges = [ {rangeStart: new Date(2003, 10, 7)} ];
```

L'exemple suivant désactive toutes les dates situées avant le 7 octobre :

```
monDC.disabledRanges = [ {rangeEnd: new Date(2002, 9, 7)} ];
```

L'exemple suivant désactive uniquement le 7 décembre :

```
monDC.disabledRanges = [ new Date(2003, 11, 7) ];
```

DateChooser.displayedMonth

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

monDC.displayedMonth

Description

Propriété : un nombre indiquant le mois à afficher. Le nombre indique un élément du tableau `monthNames`, 0 correspondant au premier mois. La valeur par défaut est le mois de la date courante.

Exemple

L'exemple suivant définit le mois affiché sur Décembre :

```
monDC.displayedMonth = 11;
```

Voir aussi

`DateChooser.displayedYear`

DateChooser.displayedYear

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monDC.displayedYear
```

Description

Propriété : un nombre à quatre chiffres indiquant l'année affichée. La valeur par défaut est l'année en cours.

Exemple

L'exemple suivant définit l'année affichée sur 2010 :

```
monDC.displayedYear = 2010;
```

Voir aussi

`DateChooser.displayedMonth`

DateChooser.firstDayOfWeek

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monDC.firstDayOfWeek
```

Description

Propriété : un nombre indiquant le jour de la semaine (0-6, 0 correspondant au premier élément du tableau `dayNames`) affiché dans la première colonne du composant DateChooser. La modification de cette propriété modifie l'ordre des colonnes des jours mais n'a aucune incidence sur l'ordre de la propriété `dayNames`. La valeur par défaut est 0 (dimanche).

Exemple

L'exemple suivant définit le premier jour de la semaine sur Lundi :

```
monDC.firstDayOfWeek = 1;
```

Voir aussi

`DateChooser.dayNames`

DateChooser.monthNames

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

monDC`.monthNames`

Description

Propriété : un tableau de chaînes indiquant les noms des mois dans la partie supérieure du composant DateChooser. La valeur par défaut est `["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"]`.

Exemple

L'exemple suivant définit les noms des mois pour l'occurrence `monDC` :

```
monDC.monthNames = ["Jan", "Feb","Mar","Apr", "May", "June","July", "Aug",
  "Sept","Oct", "Nov", "Dec"];
```

DateChooser.scroll

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

Usage 1 :

```
on(scroll){
   ...
}
```

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.scroll = function(objetEvt){
  ...
}
monDC.addEventListener("scroll", objetDécoute)
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque l'utilisateur clique sur le bouton d'un mois.

Le premier exemple d'utilisation fait appel à un gestionnaire `on()` et doit être directement associé à une occurrence de composant DateChooser. 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 stepper `monDC`, envoie « _level0.monDC » 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 (*monDC*) distribue un événement (ici, `scroll`) qui est géré par une fonction, également appelée « *gestionnaire* », sur 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 transmet automatiquement un objet événement (*objetEvt*) 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. L'objet événement de l'événement scroll comprend une propriété supplémentaire, `detail`, qui peut prendre l'une des valeurs suivantes : `nextMonth`, `previousMonth`, `nextYear`, `previousYear`.

Pour finir, vous appelez la méthode `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 événement, consultez *Objets événement*, page 592.

Exemple

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

```
form = new Object();
form.scroll = function(objEvt){
  trace(objEvt.detail);
}
monDC.addEventListener("scroll", form);
```

DateChooser.selectableRange

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

monDC.selectableRange

Description

Propriété : définit une seule date sélectionnable ou une plage de dates sélectionnables. L'utilisateur ne peut pas faire défiler l'affichage au-delà de la plage sélectionnable. La valeur de cette propriété est un objet qui comprend deux objets Date appelés `rangeStart` et `rangeEnd`. Les propriétés `rangeStart` et `rangeEnd` désignent les limites de la plage de dates sélectionnable. Si seule la propriété `rangeStart` est définie, toutes les dates situées après `rangeStart` sont activées. Si seule la propriété `rangeEnd` est définie, toutes les dates situées avant `rangeEnd` sont activées. La valeur par défaut est undefined.

Si vous ne souhaitez activer qu'un seul jour, vous pouvez utiliser un objet Date unique comme valeur de `selectableRange`.

Lorsque vous définissez une date, spécifiez une date complète. Par exemple, `new Date(2003,6,24)` plutôt que `new Date()`. Si vous ne spécifiez pas une date complète, le temps renvoie `00:00:00`.

La valeur de `DateChooser.selectedDate` est définie sur undefined si elle se situe en dehors de la plage sélectionnable.

La valeur de `DateChooser.displayedMonth` et de `DateChooser.displayedYear` est définie sur le dernier mois le plus proche dans la plage sélectionnable si le mois courant se situe en dehors de cette dernière. Par exemple, si le mois courant affiché est le mois d'août et que la plage sélectionnable est comprise entre juin 2003 et juillet 2003, le mois de juillet 2003 s'affiche.

Exemple

L'exemple suivant définit la plage sélectionnable entre le 7 mai (inclus) et le 7 juin (inclus) :

```
monDC.selectableRange = {rangeStart: new Date(2001, 4, 7), rangeEnd: new
  Date(2003, 5, 7)};
```

L'exemple suivant définit la plage sélectionnable sur les dates à compter du 7 mai (inclus) :

```
monDC.selectableRange = {rangeStart: new Date(2003, 4, 7)};
```

L'exemple suivant définit la plage sélectionnable sur les dates qui précèdent le 7 juin (jusqu'au 7 juin inclus) :

```
monDC.selectableRange = {rangeEnd: new Date(2003, 5, 7) };
```

L'exemple suivant définit la date sélectionnable uniquement au 7 juin :

```
monDC.selectableRange = new Date(2003, 5, 7);
```

DateChooser.selectedDate

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

monDC.selectedDate

Description

Propriété : un objet Date qui indique la date sélectionnée si la valeur est comprise dans les valeurs de la propriété `selectableRange`. La valeur par défaut est undefined.

La propriété `selectedDate` ne peut pas être définie dans une `disabledRange`, en-dehors d'une `selectableRange` ou sur un jour qui a été désactivé. Si la propriété `selectedDate` est définie sur l'une des dates précédentes, la valeur sera undefined.

Exemple

L'exemple suivant définit la date sélectionnée sur le 7 juin :

```
monDC.selectedDate = new Date(2003, 5, 7);
```

DateChooser.showToday

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

monDC.showToday

Description

Propriété : cette propriété définit si la date courante est mise en surbrillance. La valeur par défaut est `true`.

Exemple

L'exemple suivant désactive la mise en surbrillance de la date du jour :

```
monDC.showToday = false;
```

Composant DateField (Flash Professionnel uniquement)

Le composant DateField est un champ de texte non sélectionnable qui affiche la date accompagnée, sur la droite, d'une icône de calendrier. Si aucune date n'a été sélectionnée, le champ de texte est vide et la date du mois en cours s'affiche dans le sélecteur de dates. Lorsqu'un utilisateur clique dans le cadre de délimitation du champ de date, un sélecteur de dates présentant les dates du mois de la date sélectionnée apparaît. Lorsque le sélecteur de dates est ouvert, les utilisateurs peuvent utiliser les boutons de défilement des mois pour faire défiler les mois et les années, afin de sélectionner une date. Lorsqu'une date est sélectionnée, le sélecteur de dates se ferme.

L'aperçu en direct de DateField ne reflète pas les valeurs indiquées par le panneau de l'inspecteur des propriétés ou des composants au cours de la programmation, car il s'agit d'un composant contextuel qui n'est pas visible lors de la programmation.

Utilisation du composant DateField (Flash Professionnel uniquement)

Le composant DateField peut être utilisé là où vous souhaitez qu'un utilisateur sélectionne une date. Par exemple, vous pouvez utiliser un composant DateField dans un système de réservation d'hôtel comprenant des dates sélectionnables et des dates désactivées. Vous pouvez également utiliser le composant DateField dans une application qui affiche des événements courants, tels que des spectacles ou des réunions, lorsque l'utilisateur sélectionne une date.

Paramètres DateField

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

monthNames définit les noms des mois affichés dans la ligne d'en-tête du calendrier. La valeur est un tableau et la valeur par défaut est ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October","November", "December"].

dayNames définit les noms des jours de la semaine. La valeur est un tableau et la valeur par défaut est ["S", "M", "T", "W", "T", "F", "S"].

firstDayOfWeek indique le jour de la semaine (0-6, 0 étant le premier élément du tableau `dayNames`) affiché dans la première colonne de DateChooser. Cette propriété modifie l'ordre d'affichage des colonnes des jours.

La valeur par défaut est 0, qui correspond à "S".

disabledDays indique les jours désactivés de la semaine. Ce paramètre est un tableau qui peut comprendre un maximum de 7 valeurs. La valeur par défaut est [] (tableau vide).

showToday indique si la date du jour doit être mise en surbrillance ou non. La valeur par défaut est true.

Vous pouvez rédiger du code ActionScript pour contrôler ces options et d'autres options du composant DateField à l'aide des propriétés, méthodes et événements ActionScript. Pour plus d'informations, consultez *Classe DateField (Flash Professionnel uniquement)*, page 267.

Création d'une application avec le composant DateField

La procédure suivante explique comment ajouter un composant DateField à une application lors de la programmation. Dans cet exemple, le composant DateField autorise l'utilisateur à choisir une date dans un système de réservation de vols. Toutes les dates qui précèdent la date du jour doivent être désactivées. Une plage de 15 jours au mois de décembre doit également être désactivée afin de créer une période correspondant à des vacances. Certains vols ne sont pas disponibles les lundis, par conséquent, tous les lundis doivent être désactivés pour ces vols.

Pour créer une application avec le composant DateField, effectuez les opérations suivantes :

1. Double-cliquez sur le composant DateField dans le panneau Composants pour l'ajouter sur la scène.
2. Dans l'inspecteur des propriétés, entrez le nom d'occurrence **calendrierVols**.
3. Dans le panneau Actions, entrez le code suivant dans l'image 1 du scénario pour définir la plage des dates sélectionnables :

   ```
   calendrierVols.selectableRange = {rangeStart:new Date(2001, 9, 1),
     rangeEnd:new Date(2003, 11, 1)};
   ```

   Ce code affecte une valeur à la propriété `selectableRange` dans un objet ActionScript qui contient deux objets Date avec les noms de variables `rangeStart` et `rangeEnd`. Il définit les extrémités inférieure et supérieure de la plage dans laquelle l'utilisateur peut choisir une date.
4. Dans le panneau Actions, entrez le code suivant sur l'image 1 du scénario pour définir les plages de dates désactivées, une en décembre et une pour toutes les dates précédant la date actuelle.

   ```
   calendrierVols.disabledRanges = [{rangeStart: new Date(2003, 11, 15),
     rangeEnd: new Date(2003, 11, 31)}, {rangeEnd: new Date(2003, 6, 16)}];
   ```
5. Dans le panneau Actions, entrez le code suivant dans l'image 1 du scénario pour désactiver les lundis :

   ```
   calendrierVols.disabledDays=[1];
   ```
6. Choisissez Contrôle > Tester l'animation.

Personnalisation du composant DateField (Flash Professionnel uniquement)

Vous pouvez transformer un composant DateField horizontalement au cours de la programmation et à l'exécution. Lors de la programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez la méthode `setSize()` (consultez `UIObject.setSize()`). La définition de la largeur ne modifie pas les dimensions de DateChooser dans le composant DateField. Vous pouvez néanmoins utiliser la propriété `pullDown` pour accéder au composant DateChooser et définir ses dimensions.

Utilisation des styles avec le composant DateField

Vous pouvez définir les propriétés des styles pour modifier l'apparence d'une occurrence de champ de date. Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, consultez *Utilisation des styles pour personnaliser la couleur et le texte des composants*, page 29.

Un composant DateField supporte les styles de halo suivants :

Style	Description
`themeColor`	La couleur de l'éclat pour les dates survolées et sélectionnées. 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"`.

Utilisation des enveloppes avec le composant DateField

Le composant DateField utilise des enveloppes pour représenter les états visuels de l'icône contextuelle. Pour appliquer une enveloppe à l'icône contextuelle au cours de la programmation, modifiez les symboles d'enveloppe dans le dossier Flash UI Components 2/Themes/MMDefault/DateField Elements skins states de la bibliothèque de l'un des fichiers FLA de thèmes. Pour plus d'informations, consultez *A propos de l'application des enveloppes aux composants*, page 38.

Dans ce composant, vous pouvez uniquement envelopper le bouton de l'icône contextuelle. Un composant DateField utilise les propriétés d'enveloppe suivantes pour envelopper dynamiquement l'icône contextuelle :

Propriété	Description
`openDateUp`	Etat Relevé de l'icône contextuelle.
`openDateDown`	Etat Enfoncé de l'icône contextuelle.
`openDateOver`	Etat Survolé de l'icône contextuelle.
`openDateDisabled`	Etat Désactivé de l'icône contextuelle.

Classe DateField (Flash Professionnel uniquement)

Héritage UIObject > UIComponent > ComboBase > DateField

Nom de classe ActionScript mx.controls.DateField

Les propriétés de la classe DateField vous permettent d'accéder à la date sélectionnée et au mois et à l'année affichés. Vous pouvez également définir les noms des jours et des mois, indiquer les dates désactivées et les dates sélectionnables, définir le premier jour de la semaine et indiquer si la date courante doit être mise en surbrillance.

La définition d'une propriété de la classe DateField 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.DateField.version);
```

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

Méthodes de la classe DateField

Méthode	Description
`DateField.close()`	Ferme le sous-composant DateChooser contextuel.
`DateField.open()`	Ouvre le sous-composant DateChooser contextuel.

Hérite de toutes les méthodes des classes *UIObject* et *UIComponent*.

Propriétés de la classe DateField

Propriété	Description
`DateField.dateFormatter`	Fonction qui formate la date à afficher dans le champ de texte.
`DateField.dayNames`	Un tableau indiquant les noms des jours de la semaine.
`DateField.disabledDays`	Un tableau indiquant les jours de la semaine qui sont désactivés pour toutes les dates applicables dans le sélecteur de dates.
`DateField.disabledRanges`	Une plage de dates désactivée ou une seule date désactivée.
`DateField.displayedMonth`	Un nombre indiquant un élément du tableau `monthNames` à afficher dans le sélecteur de dates.
`DateField.displayedYear`	Un nombre indiquant l'année à afficher.
`DateField.firstDayOfWeek`	Un nombre indiquant un élément du tableau `dayNames` à afficher dans la première colonne du sélecteur de dates.
`DateField.monthNames`	Un tableau de chaînes indiquant les noms des mois.
`DateField.pullDown`	Référence au sous-composant DateChooser. Cette propriété est en lecture seule.
`DateField.selectableRange`	Une seule date sélectionnable ou une plage de dates sélectionnables.
`DateField.selectedDate`	Un objet Date indiquant la date sélectionnée.
`DateField.showToday`	Une valeur booléenne indiquant si la date courante est mise en surbrillance.

Hérite de toutes les propriétés des classes *UIObject* et *UIComponent*.

Evénements de la classe DateField

Evénement	Description
`DateField.change`	Diffusé lorsqu'une date est sélectionnée.
`DateField.close`	Diffusé lors de la fermeture du sous-composant DateChooser.
`DateField.open`	Diffusé lors de l'ouverture du sous-composant DateChooser.
`DateField.scroll`	Diffusé lorsque l'utilisateur clique sur les boutons des mois.

Hérite de tous les événements des classes *UIObject* et *UIComponent*.

DateField.change

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

Usage 1 :

```
on(change){
   ...
}
```

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.change = function(objetEvt){
   ...
}
monDF.addEventListener("change", objetDécoute)
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsqu'une date est sélectionnée.

Le premier exemple d'utilisation fait appel à un gestionnaire `on()` et doit être directement associé à une occurrence de composant DateField. 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 champ de date `monDF`, envoie « _level0.monDF » 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 (*monDF*) distribue un événement (ici, `change`) qui est géré par une fonction, également appelée « *gestionnaire* », sur 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 transmet automatiquement un objet événement (*objetEvt*) 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 événement, consultez *Objets événement*, page 592.

Exemple

Cet exemple, écrit sur une image du scénario, envoie un message au panneau de sortie lorsqu'un champ de date nommé `monDF` 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 `objEvt`, pour générer un message. La propriété `target` d'un objet événement est le composant ayant généré l'événement, dans cet exemple, `monDF`. L'utilisateur accède à la propriété `DateField.selectedDate` à partir de la propriété `target` de l'objet événement. La dernière ligne appelle la méthode `UIEventDispatcher.addEventListener()` depuis `monDF` et lui transmet l'événement `change` et l'objet d'écoute `form` comme paramètres, comme dans l'exemple suivant :

```
form.change = function(objEvt){
    trace("date sélectionnée " +  eventObj.target.selectedDate) ;
}
monDF.addEventListener("change", form);
```

DateField.close()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monDF.close()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : ferme le menu contextuel.

Exemple

Le code suivant ferme le composant DateChooser contextuel de l'occurrence de champ de date `monDF` :

```
monDF.close();
```

DateField.close

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

Usage 1 :

```
on(close){
  ...
}
```

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.close = function(objetEvt){
  ...
}
monDF.addEventListener("close", objetDécoute)
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque le sous-composant DateChooser est fermé après qu'un utilisateur a cliqué en dehors de l'icône ou a choisi une date.

Le premier exemple d'utilisation fait appel à un gestionnaire `on()` et doit être directement associé à une occurrence de composant DateField. 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 champ de date `monDF`, envoie « _level0.monDF » au panneau de sortie :

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*monDF*) distribue un événement (ici, `close`) qui est géré par une fonction, également appelée un *gestionnaire*, 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 transmet automatiquement un objet événement (*objetEvt*) 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 événement, consultez *Objets événement*, page 592.

Exemple

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

```
form.close = function(eventObj){
  trace("Menu déroulant fermé" + eventObj.target.selectedDate);
}
monDF.addEventListener("close", form);
```

DateField.dateFormatter

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

monDF.dateFormatter

Description

Propriété : fonction qui formate la date à afficher dans le champ de texte. La fonction doit recevoir un objet Date comme paramètre et renvoyer une chaîne au format à afficher.

Exemple

L'exemple suivant définit la fonction pour renvoyer le format de la date à afficher :

```
monDF.dateFormatter = function(d:Date){
  return d.getFullYear()+"/ "+(d.getMonth()+1)+"/ "+d.getDate();
};
```

DateField.dayNames

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

monDF.dayNames

Description

Propriété : un tableau contenant les noms des jours de la semaine. Dimanche est le premier jour (à l'emplacement d'index 0) et les autres noms de jours suivent dans l'ordre. La valeur par défaut est `["S", "M", "T", "W", "T", "F", "S"]`.

Exemple

L'exemple suivant change la valeur du cinquième jour de la semaine (jeudi) en la faisant passer de "T" à "R" :

```
monDF.dayNames[4] = "R";
```

DateField.disabledDays

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monDF.disabledDays
```

Description

Propriété : un tableau indiquant les jours désactivés de la semaine. Toutes les dates du mois qui correspondent au jour spécifié sont désactivées. Les éléments de ce tableau peuvent avoir des valeurs comprises entre 0 (dimanche) et 6 (samedi). La valeur par défaut est `[]` (tableau vide).

Exemple

L'exemple suivant désactive les dimanches et samedis pour que les utilisateurs puissent uniquement choisir des jours de semaine :

```
monDF.disabledDays = [0, 6];
```

DateField.disabledRanges

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monDF.disabledRanges
```

Description

Propriété : désactive un seul jour ou une plage de jours. Cette propriété est un tableau d'objets. Chaque objet du tableau doit être soit un objet Date spécifiant un seul jour à désactiver, soit un objet contenant une des propriétés `rangeStart` ou `rangeEnd`, voire les deux, dont les valeurs doivent être un objet Date. Les propriétés `rangeStart` et `rangeEnd` décrivent les limites de la plage de dates. Si l'une ou l'autre des propriétés est omise, l'étendue est illimitée dans le sens correspondant.

La valeur par défaut de `disabledRanges` est undefined.

Lorsque vous définissez des dates pour la propriété `disabledRanges`, spécifiez des dates complètes. Par exemple, `new Date(2003,6,24)` plutôt que `new Date()`. Si vous ne spécifiez pas une date complète, le temps renvoie `00:00:00`.

Exemple

L'exemple suivant définit un tableau comprenant des objets Date `rangeStart` et `rangeEnd` qui désactivent les dates comprises entre le 7 mai et le 7 juin :

```
monDF.disabledRanges = [ {rangeStart: new Date(2003, 4, 7), rangeEnd: new
  Date(2003, 5, 7)}];
```

L'exemple suivant désactive toutes les dates situées après le 7 novembre :

```
monDF.disabledRanges = [ {rangeStart: new Date(2003, 10, 7)} ];
```

L'exemple suivant désactive toutes les dates situées avant le 7 octobre :

```
monDF.disabledRanges = [ {rangeEnd: new Date(2002, 9, 7)} ];
```

L'exemple suivant désactive uniquement le 7 décembre :

```
monDF.disabledRanges = [ new Date(2003, 11, 7) ];
```

DateField.displayedMonth

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

monDF.displayedMonth

Description

Propriété : un nombre indiquant le mois à afficher. Le nombre indique un élément du tableau `monthNames`, 0 correspondant au premier mois. La valeur par défaut est le mois de la date courante.

Exemple

L'exemple suivant définit le mois affiché sur Décembre :

```
monDF.displayedMonth = 11;
```

Voir aussi

DateField.displayedYear

DateField.displayedYear

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

monDF.displayedYear

Description

Propriété : nombre indiquant l'année qui est affichée. La valeur par défaut est l'année en cours.

Exemple

L'exemple suivant définit l'année affichée sur 2010 :

```
monDF.displayedYear = 2010;
```

Voir aussi

DateField.displayedMonth

DateField.firstDayOfWeek

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

monDF.firstDayOfWeek

Description

Propriété : nombre indiquant le jour de la semaine (0-6, 0 étant le premier élément du tableau `dayNames`) qui est affiché dans la première colonne du composant DateField. La modification de cette propriété modifie l'ordre des colonnes des jours mais n'a aucune incidence sur l'ordre de la propriété `dayNames`. La valeur par défaut est 0 (dimanche).

Exemple

L'exemple suivant définit le premier jour de la semaine sur Lundi :

```
monDF.firstDayOfWeek = 1;
```

Voir aussi

DateField.dayNames

DateField.monthNames

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

monDF.monthNames

Description

Propriété : tableau de chaînes indiquant les noms des mois en haut du composant DateField. La valeur par défaut est ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"].

Exemple

L'exemple suivant définit les noms de mois pour l'occurrence monDF :

```
monDF.monthNames = ["Jan", "Fév","Mars","Avr", "Mai", "Juin","Juil", "Août",
   "Sept","Oct", "Nov", "Déc"];
```

DateField.open()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

monDF.open()

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : ouvre le sous-composant DateChooser contextuel.

Exemple

Le code suivant ouvre le sous-composant DateChooser contextuel de l'occurrence df :

```
df.open();
```

DateField.open

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

Usage 1 :

```
on(open){
   ...
}
```

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.open = function(ObjetEvt){
   ...
}
monDF.addEventListener("open", objetDécoute)
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsqu'un sous-composant DateChooser est ouvert après qu'un utilisateur a cliqué sur l'icône.

Le premier exemple d'utilisation fait appel à un gestionnaire `on()` et doit être directement associé à une occurrence de composant DateField. 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 champ de date `monDF`, envoie « _level0.monDF » au panneau de sortie :

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

Le deuxième exemple d'utilisation fait appel à un modèle d'événement dispatcher/écouteur. Une occurrence de composant (*`monDF`*) distribue un événement (ici, `open`) qui est géré par une fonction, également appelée un *gestionnaire*, 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

Exemple

Cet exemple, écrit sur une image du scénario, envoie un message au panneau de sortie lorsqu'un stepper nommé `monDF` est ouvert. La première ligne de code crée un objet d'écoute intitulé `form`. La deuxième ligne définit une fonction pour l'événement `open` sur l'objet d'écoute. La fonction comporte une action `trace` qui utilise l'objet événement automatiquement transmis à cette fonction, ici `objEvt`, pour générer un message. La propriété `target` d'un objet événement est le composant ayant généré l'événement, dans cet exemple, `monDF`. L'utilisateur accède à la propriété `DateField.selectedDate` à partir de la propriété `target` de l'objet événement. La dernière ligne appelle la méthode `UIEventDispatcher.addEventListener()` depuis `monDF` et lui transmet l'événement `open` et l'objet d'écoute `form` comme paramètres, comme dans l'exemple suivant :

```
form.open = function(objEvt){
    trace("Le menu est ouvert et la date sélectionnée est " +
  ojvEvnt.target.selectedDate) ;
}
monDF.addEventListener("open", form);
```

DateField.pullDown

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

monDF.pullDown

Description

Propriété (lecture seule) : référence au composant DateChooser contenu dans le composant DateField. Le sous-composant DateChooser est instancié lorsqu'un utilisateur clique sur le composant DateField. Néanmoins, si la propriété `pullDown` est référencée avant que l'utilisateur ne clique sur le composant, le sous-composant DateChooser est instancié puis masqué.

Exemple

L'exemple suivant définit la visibilité du sous-composant DateChooser sur `false` puis définit les dimensions du sous-composant DateChooser à 300 pixels de hauteur et 300 pixels de largeur :

```
monDF.pullDown._visible = false;
monDF.pullDown.setSize(300,300);
```

DateField.scroll

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

Usage 1 :

```
on(scroll){
   ...
}
```

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.scroll = function(objetEvt){
   ...
}
monDF.addEventListener("scroll", objetDécoute)
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque l'utilisateur clique sur le bouton d'un mois.

Le premier exemple d'utilisation fait appel à un gestionnaire `on()` et doit être directement associé à une occurrence de composant DateField. 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 champ de date `monDF`, envoie « _level0.monDF » 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 (*`monDF`*) distribue un événement (ici, `scroll`) qui est géré par une fonction, également appelée un *gestionnaire*, 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 transmet automatiquement un objet événement (*`objetEvt`*) 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. L'objet événement de l'événement scroll comprend une propriété supplémentaire, `detail`, qui peut prendre l'une des valeurs suivantes : `nextMonth`, `previousMonth`, `nextYear`, `previousYear`.

Pour finir, vous appelez la méthode `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 événement, consultez *Objets événement*, page 592.

Exemple

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

```
form = new Object();
form.scroll = function(objEvt){
  trace(objEvt.detail);
}
monDF.addEventListener("scroll", form);
```

DateField.selectableRange

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

monDF`.selectableRange`

Description

Propriété : définit une seule date sélectionnable ou une plage de dates sélectionnables. La valeur de cette propriété est un objet qui comprend deux objets Date appelés `rangeStart` et `rangeEnd`. Les propriétés `rangeStart` et `rangeEnd` désignent les limites de la plage de dates sélectionnable. Si seule la propriété `rangeStart` est définie, toutes les dates situées après `rangeStart` sont activées. Si seule la propriété `rangeEnd` est définie, toutes les dates situées avant `rangeEnd` sont activées. La valeur par défaut est undefined.

Si vous ne souhaitez activer qu'un seul jour, vous pouvez utiliser un objet Date unique comme valeur de `selectableRange`.

Lorsque vous définissez une date, spécifiez une date complète. Par exemple, `new Date(2003,6,24)` plutôt que `new Date()`. Si vous ne spécifiez pas une date complète, le temps renvoie `00:00:00`.

La valeur de `DateField.selectedDate` est définie sur undefined si elle se situe en dehors de la plage sélectionnable.

La valeur de `DateField.displayedMonth` et de `DateField.displayedYear` est définie sur le dernier mois le plus proche dans la plage sélectionnable si le mois courant se situe en dehors de cette dernière. Par exemple, si le mois courant affiché est le mois d'août et que la plage sélectionnable est comprise entre juin 2003 et juillet 2003, le mois de juillet 2003 s'affiche.

Exemple

L'exemple suivant définit la plage sélectionnable entre le 7 mai (inclus) et le 7 juin (inclus) :

```
monDF.selectableRange = {rangeStart: new Date(2001, 4, 7), rangeEnd: new
  Date(2003, 5, 7)};
```

L'exemple suivant définit la plage sélectionnable sur les dates à compter du 7 mai (inclus) :

```
monDF.selectableRange = {rangeStart: new Date(2003, 4, 7)};
```

L'exemple suivant définit la plage sélectionnable sur les dates qui précèdent le 7 juin (jusqu'au 7 juin inclus) :

```
monDF.selectableRange = {rangeEnd: new Date(2003, 5, 7) };
```

L'exemple suivant définit la date sélectionnable uniquement au 7 juin :

```
monDF.selectableRange = new Date(2003, 5, 7);
```

DateField.selectedDate

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

monDF.selectedDate

Description

Propriété : un objet Date qui indique la date sélectionnée si la valeur est comprise dans les valeurs de la propriété `selectableRange`. La valeur par défaut est undefined.

Exemple

L'exemple suivant définit la date sélectionnée sur le 7 juin :

```
monDF.selectedDate = new Date(2003, 5, 7);
```

DateField.showToday

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

monDF.showToday

Description

Propriété : cette propriété définit si la date courante est mise en surbrillance. La valeur par défaut est `true`.

Exemple

L'exemple suivant désactive la mise en surbrillance de la date du jour :

```
monDF.showToday = false;
```

Classe DepthManager

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 version 79.

Edition

Flash MX 2004.

Usage

```
occurrenceClip.createChildAtDepth(nomLiaison, indicateurProfondeur[, objInit])
```

Paramètres

nomLiaison Identificateur de liaison. Ce paramètre est une chaîne.

indicateurProfondeur L'une des valeurs suivantes : `DepthManager.kTop`, `DepthManager.kBottom`, `DepthManager.kTopmost`, `DepthManager.kNotopmost`. Tous les indicateurs de profondeur sont des propriétés statiques de la classe DepthManager. Vous devez 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 *indicateurProfondeur*.

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 version 79.

Edition

Flash MX 2004 et Flash MX Professionnel 2004.

Usage

```
occurrenceClip.createClassChildAtDepth( nomClasse, indicateurProfondeur,
  objInit )
```

Paramètres

nomClasse Nom de classe.

indicateurProfondeur L'une des valeurs suivantes : `DepthManager.kTop`, `DepthManager.kBottom`, `DepthManager.kTopmost`, `DepthManager.kNotopmost`. Tous les indicateurs de profondeur sont des propriétés statiques de la classe DepthManager. Vous devez 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 *indicateurProfondeur*.

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 version 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 indicateurs 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 version 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 indicateurs 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 version 79.

Edition

Flash MX 2004 et Flash MX Professionnel 2004.

Usage

```
occurrenceDeClip.setDepthAbove(occurrence)
```

Paramètres

occurrence 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 *occurrence*.

DepthManager.setDepthBelow()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004 et Flash MX Professionnel 2004.

Usage

```
occurrenceDeClip.setDepthBelow(occurrence)
```

Paramètres

occurrence 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 *occurrence*.

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 version 79.

Edition

Flash MX 2004 et Flash MX Professionnel 2004.

Usage

```
occurrenceDeClip.setDepthTo(profondeur)
```

Paramètres

profondeur Niveau de profondeur.

Renvoie

Rien.

Description

Méthode : définit la profondeur de *occurrenceDeClip* 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 Guide de référence 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.

FocusManager reconnaît les groupes de boutons radio (ceux dont la propriété `RadioButton.groupName` est définie) et règle le focus sur l'occurrence du groupe dont la propriété `selected` 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

Le Gestionnaire de focus n'affecte pas automatiquement de focus à un composant. Vous devez écrire un script qui appelle `FocusManager.setFocus()` sur un composant si vous souhaitez qu'un composant reçoive le focus lors du chargement d'une application.

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 :

```
commentaire.tabIndex = 1;
boutonOK.tabIndex = 2;
```

Vous pouvez également utiliser le panneau Accessibilité pour affecter une valeur d'indexation.

Si aucune valeur d'indexation n'est définie sur la scène, FocusManager utilise l'ordre z. L'ordre z est principalement défini par l'ordre dans lequel les composants sont placés sur la scène. Vous pouvez néanmoins utiliser les commandes Modifier/Réorganiser/Mettre au premier plan/Mettre à l'arrière-plan pour déterminer l'ordre z final.

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 programmation 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 :

```
commentaire.tabIndex = 1;
boutonOK.tabIndex = 2;
focusManager.setFocus(comment);
focusManager.defaultPushButton = okButton;
lo = new Object();
lo.click = function(evt){
  trace(evt.target + " a été 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

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 version 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;
```

Voir aussi

FocusManager.defaultPushButtonEnabled,
FocusManager.sendDefaultPushButtonEvent()

FocusManager.defaultPushButtonEnabled

Disponibilité

Flash Player 6 version 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 version 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 de 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 version 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` :

```
if (focusManager.getFocus() == maSaisieDeTexte)
{
  focusManager.setFocus(monBoutonOK);
}
```

Voir aussi

`FocusManager.setFocus()`

FocusManager.nextTabIndex

Disponibilité

Flash Player 6 version 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;
```

Voir aussi

`UIComponent.tabIndex`

FocusManager.sendDefaultPushButtonEvent()

Disponibilité

Flash Player 6 version 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 ») :

```
nom_txt.tabIndex = 1;
motDePasse_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){
    nom_txt.text = "Lucie";
    motDePasse_txt.text = "secret";
    focusManager.sendDefaultPushButtonEvent();
} else {
    nom_txt.text = "";
    motDePasse_txt.text = "";
  }
}
chb.addEventListener("click", objChb);

submitObj = new Object();
submitObj.click = function(o){
  if (motDePasse_txt.text != "secret"){
    trace("erreur de soumission");
} else {
```

```
        trace("sendDefaultPushButtonEvent a abouti !");
    }
}
submit_ib.addEventListener("click", submitObj);
```

Voir aussi

FocusManager.defaultPushButton, FocusManager.sendDefaultPushButtonEvent()

FocusManager.setFocus()

Disponibilité

Flash Player 6 version 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);
```

Voir aussi

FocusManager.getFocus()

Classe Form (Flash Professionnel uniquement)

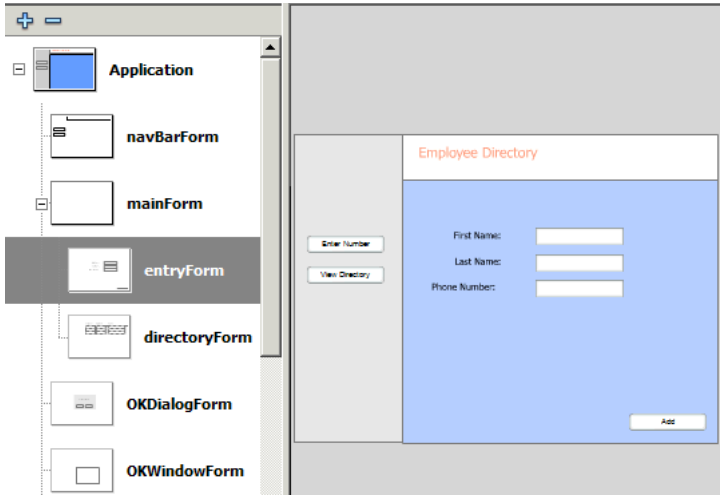
Héritage UIObject > UIComponent > View > Loader > Screen > Form

Nom de classe ActionScript mx.screens.Form

La classe Form fournit le comportement à l'exécution des formulaires que vous créez dans le panneau Contour de l'écran dans Flash MX Professionnel 2004. Pour un aperçu de l'utilisation des écrans, consultez « Utilisation des écrans (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Utilisation de la classe Form (Flash Professionnel uniquement)

Les formulaires fonctionnent à la fois comme des conteneurs pour les objets graphiques (des éléments d'interface utilisateur dans une application, par exemple) et comme des états d'application. Vous pouvez utiliser le panneau Contour de l'écran pour visualiser les différents états d'une application que vous créez, où chaque formulaire est un état d'application différent. Par exemple, l'illustration suivante montre le panneau Contour de l'écran pour un exemple d'application conçu à l'aide de formulaires.



Vue du Contour de l'écran d'un exemple d'application de formulaires.

Cette illustration montre le contour d'un exemple d'application appelé « Employee Directory » (Répertoire des employés), qui comprend plusieurs formulaires. Le formulaire nommé « entryForm » (formulaire d'entrée, sélectionné dans l'illustration ci-dessus) contient plusieurs objets d'interface utilisateur, y compris des champs de saisie de texte, des étiquettes et un bouton-poussoir. Le développeur peut facilement présenter ce formulaire à l'utilisateur en faisant basculer sa visibilité, (à l'aide de la propriété `Form.visible`) tout en faisant basculer la visibilité des autres formulaires simultanément.

Vous pouvez également associer des comportements et des commandes aux formulaires à l'aide du panneau Comportements (Fenêtre > Panneaux de développement > Comportements). Pour plus d'informations sur l'ajout de transitions et de commandes aux écrans, consultez « Création de commandes et de transitions pour les écrans à l'aide des comportements (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Etant donné que la classe Form étend la classe Loader, vous pouvez facilement charger du contenu externe (un fichier SWF ou JPEG) dans un formulaire. Par exemple, le contenu d'un formulaire peut être un fichier SWF séparé, pouvant lui-même contenir des formulaires. De cette façon, vous pouvez modulariser vos applications de formulaires, ce qui facilite la gestion des applications et réduit le temps de téléchargement initial. Pour plus d'informations, consultez *Chargement de contenu externe dans des écrans (Flash Professionnel uniquement)*, page 478.

Paramètres de l'objet Form

Vous pouvez définir les paramètres de programmation suivants pour chaque occurrence de l'objet Form dans l'inspecteur des propriétés ou des composants :

autoload Indique si le contenu spécifié par le paramètre `contentPath` doit être chargé automatiquement (true) ou s'il faut attendre que la méthode `Loader.load()` soit appelée (false). La valeur par défaut est true.

contentPath Spécifie le contenu du formulaire. Il peut s'agir de l'identificateur de liaison d'un clip ou d'une URL absolue ou relative d'un fichier SWF ou JPG à charger dans la diapositive. Par défaut, le contenu chargé est coupé pour être adapté à la diapositive.

visible Spécifie si le formulaire est visible (true) ou non (false) lors de son premier chargement.

Méthodes de la classe Form

Méthode	Description
`Form.getChildForm()`	Renvoie le formulaire enfant à un index spécifié.

Hérite de toutes les méthodes des classes *UIObject*, *UIComponent*, View, du *Composant Loader* et de la *Classe Screen (Flash Professionnel uniquement)*.

Propriétés de la classe Form

Propriété	Description
`Form.currentFocusedForm`	Renvoie le formulaire comportant le plus de feuilles qui contient le focus global actuel.
`Form.indexInParentForm`	Renvoie l'index (basé sur zéro) de ce formulaire dans la liste des sous-formulaires de son parent.
`Form.visible`	Spécifie si le formulaire est visible lorsque son formulaire, sa diapositive, son clip ou son fichier SWF parent est visible.
`Form.numChildForms`	Renvoie le nombre de formulaires enfants que ce formulaire contient.
`Form.parentIsForm`	Indique si l'objet parent de ce formulaire est également un formulaire.
`Form.rootForm`	Renvoie la racine de l'arborescence ou sous-arborescence du formulaire qui contient le formulaire.

Hérite de toutes les propriétés des composants *UIObject*, *UIComponent*, View, du *Composant Loader* et de la *Classe Screen (Flash Professionnel uniquement)*.

Form.currentFocusedForm

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

```
mx.screens.Form.currentFocusedForm
```

Description

Propriété (lecture seule) : renvoie l'objet Form contenant le focus global actuel. Le focus actuel peut être sur le formulaire lui-même ou sur un clip, un objet texte ou un composant dans le formulaire. Peut être null s'il n'y a aucun focus actuel.

Exemple

Le code suivant, associé à un bouton (masqué), affiche le nom du formulaire ayant le focus actuel.

```
trace("Le formulaire ayant le focus actuel est : " +
  mx.screens.Form.currentFocusedForm);
```

Form.getChildForm()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

```
monFormulaire.getChildForm(indexEnfant)
```

Paramètres

indexEnfant Nombre indiquant l'index (basé sur zéro) du formulaire enfant à renvoyer.

Renvoie

Un objet Form.

Description

Méthode : renvoie le formulaire enfant de *monFormulaire* dont l'index est *indexEnfant*.

Exemple

L'exemple suivant affiche dans le panneau de sortie les noms de tous les objets Form enfants qui appartiennent à l'objet racine Form nommé `Application`.

```
for (var i:Number = 0; i < _root.Application.numChildForms; i++) {
    var formEnfant:mx.screens.Form = _root.Application.getChildForm(i);
    trace(formEnfant._name);
}
```

Voir aussi

`Form.numChildForms`

Form.indexInParentForm

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`monFormulaire``.indexInParentForm`

Description

Propriété (lecture seule) : contient l'index (basé sur zéro) de *`monFormulaire`* dans la liste des formulaires enfants de son parent. Si l'objet parent de *`monFormulaire`* est un écran et non un formulaire (par exemple, une diapositive), alors `indexInParentForm` est toujours 0.

Exemple

```
var monIndex:Number = monFormulaire.indexInParent;
if (monFormulaire == monFormulaire._parent.getChildForm(monIndex)) {
  trace("Je suis au bon endroit");
}
```

Voir aussi

`Form.getChildForm()`

Form.numChildForms

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`monFormulaire``.numChildForms`

Description

Propriété (lecture seule) : renvoie le nombre de formulaires enfants contenus dans *`monFormulaire`*. Cette propriété n'inclut aucune diapositive contenue dans *`monFormulaire`*, uniquement des formulaires.

Exemple

Le code suivant itère sur tous les formulaires enfants contenus dans `monFormulaire` et affiche leurs noms dans le panneau de sortie.

```
var combienDEnfants:Number = monFormulaire.numChildForms;
for(i=0; i<combienDEnfants; i++) {
  var formEnfant = monFormulaire.getChildForm(i);
  trace(formEnfant._name);
}
```

Voir aussi

`Form.getChildForm()`

Form.parentIsForm

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`monFormulaire``.parentIsForm`

Description

Propriété (lecture seule) : Renvoie une valeur booléenne (`true` ou `false`) indiquant si l'objet parent du formulaire est également un formulaire (`true`) ou non (`false`). Si la valeur est false, *`monFormulaire`* est à la racine de sa hiérarchie de formulaires.

Exemple

```
if (monFormulaire.parentIsForm) {
  trace("J'ai "+monFormulaire._parent.numChildScreens+" écrans frères");
} else {
  trace("Je suis le formulaire racine et je n'ai pas de frères");
}
```

Form.rootForm

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`monFormulaire``.rootForm`

Description

Propriété (lecture seule) : renvoie le formulaire au sommet de la hiérarchie de formulaires contenant *`monFormulaire`*. Si *`monFormulaire`* est contenu dans un objet qui n'est pas un formulaire (c'est-à-dire une diapositive), cette propriété renvoie alors *`monFormulaire`*.

Exemple

Dans l'exemple suivant, une référence au formulaire racine de `monFormulaire` est placée dans une variable nommée `racine`. Si la valeur affectée à `racine` fait référence à `monFormulaire`, alors `monFormulaire` est au sommet de son arborescence de formulaires.

```
var racine:mes.écrans.Form = monFormulaire.rootForm;
if(rootForm == monFormulaire) {
  trace("monFormulaire est le formulaire de premier niveau dans son
  arborescence");
}
```

Form.visible

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`monFormulaire``.visible`

Description

Propriété : détermine si *`monFormulaire`* est visible lorsque son formulaire, sa diapositive, son clip ou son animation parent est visible. Vous pouvez également définir cette propriété à l'aide de l'inspecteur des propriétés dans l'environnement de programmation de Flash.

Lorsque cette propriété est définie sur `true`, *`monFormulaire`* reçoit un événement `reveal`. Lorsqu'elle est définie sur `false`, *`monFormulaire`* reçoit un événement `hide`. Vous pouvez associer des transitions aux formulaires exécutés lorsqu'un formulaire reçoit l'un de ces événements. Pour plus d'informations sur l'ajout de transitions aux écrans, consultez « Création de commandes et de transitions pour les écrans à l'aide des comportements (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Exemple

Le code suivant, associé à une occurrence du composant Button, définit sur `false` la propriété `visible` du formulaire contenant le bouton.

```
on(click) {
  _parent.visible = true;
}
```

Composant Label

Un composant d'étiquette est une ligne unique de texte. Vous pouvez spécifier qu'une étiquette devra être formatée en 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. Le paramètre autoSize n'est pas supporté dans l'aperçu en direct.

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.

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 programmation 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 en 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 des composants, entrez **Date d'expiration** pour le paramètre d'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 programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. Vous pouvez également définir le paramètre de programmation autoSize ; la définition de ce paramètre ne change pas le cadre de délimitation dans l'aperçu en direct, mais l'étiquette est redimensionnée. Pour plus d'informations, consultez *Paramètres Label*, page 301. 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 29.

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 38.

Classe Label

Héritage UIObject > Label

Nom de classe ActionScript mx.controls.Label

Lors de l'exécution, les propriétés de la classe Label permettent de spécifier du texte pour l'étiquette, d'indiquer si le texte peut être formaté en 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 en 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 version 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 version 79.

Edition

Flash MX 2004.

Usage

occurrenceEtiquette.html

Description

Propriété : valeur booléenne indiquant si l'étiquette peut être formatée en 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 en 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 version 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 27 ou *Classe FocusManager*, page 287.

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.

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. Imaginons qu'il ait le choix parmi 30 articles. Il 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 programmation et l'exécution. Lors de la programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez la méthode `List.setSize()` (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 490.

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.
`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.

Style	Description
selectionDuration	Longueur des transitions lors de la sélection des éléments.
useRollOver	Détermine si le survol d'une ligne active sa mise en surbrillance.

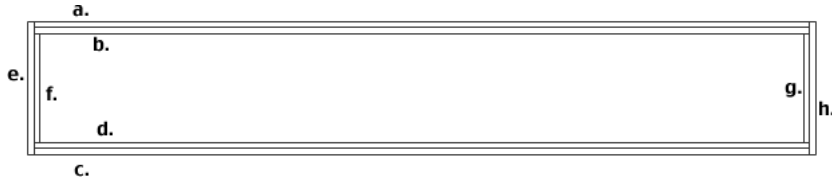
Les composants List utilisent également les propriétés de style du composant Label (voir *Utilisation des styles avec le composant Label*, page 302), du composant ScrollPane (voir *Composant ScrollPane*, page 490) et du composant RectBorder.

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 490. 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

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, taille, 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` (voir `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 27.

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);`.

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.

Méthode	Description
`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 des classes *UIObject* et *UIComponent*.

Propriétés de la classe List

Propriété	Description
`List.cellRenderer`	Affecte la classe ou le symbole à utiliser pour afficher chaque ligne de la liste.
`List.dataProvider`	Source des éléments de la liste.
`List.hPosition`	Position horizontale de la liste.
`List.hScrollPolicy`	Indique si la barre de défilement horizontale est affichée (`"on"`) ou non (`"off"`).
`List.iconField`	Champ situé 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.
`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.

Propriété	Description
`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 des classes *UIObject* et *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 des classes *UIObject* et *UIComponent*.

List.addItem()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

```
OccurrenceListe.addItem(label[, data])
OccurrenceListe.addItem(objetElément)
```

Paramètres

label Chaîne indiquant l'étiquette du nouvel élément.

data 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 version 79.

Edition

Flash MX 2004.

Usage

```
OccurrenceListe.addItemAt(index, label [,data])
OccurrenceListe.addItemAt(index, objetElément)
```

Paramètres

`label` Chaîne indiquant l'étiquette du nouvel élément.

`data` 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 version 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. Toutes les classes utilisées pour cette propriété doivent implémenter *API CellRenderer*, page 84.

Exemple

L'exemple suivant utilise un identificateur de liaison pour définir un nouvel objet cellRenderer :

```
maListe.cellRenderer = "ComboBoxCell";
```

List.change

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.change = function(objetEvt){
  // 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, également appelée un *gestionnaire*, 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 transmet automatiquement un objet événement (*objetEvt*) 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 événement, consultez *Objets événement*, page 592.

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(objEvt){
  trace("Valeur passée à " + objEvt.target.value);
}
maListe.addEventListener("change", form);
```

Voir aussi

`UIEventDispatcher.addEventListener()`

List.dataProvider

Disponibilité

Flash Player 6 version 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 implémentant l'interface DataProvider. La valeur par défaut est []. Pour plus d'informations sur l'interface DataProvider, consultez *API DataProvider*, page 196.

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;

for (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 version 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. Undefined 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 version 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 posDéfil = 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 version 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;
```

Voir aussi

`List.hPosition`, `List.maxHPosition`

List.iconField

Disponibilité

Flash Player 6 version 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"
```

Voir aussi

List.iconFunction

List.iconFunction

Disponibilité

Flash Player 6 version 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 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.itemRollOut = function(ObjetEvt){
  // 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, également appelée un *gestionnaire*, 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 transmet automatiquement un objet événement (*objetEvt*) 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 événement, consultez *Objets événement*, page 592.

Exemple

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

```
form.itemRollOut = function (objEvt) {
  trace("Item #" + objetEvt.index + " a été survolé.");
}
maListe.addEventListener("itemRollOut", form);
```

Voir aussi

List.itemRollOver

List.itemRollOver

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.itemRollOver = function(objetEvt){
  // 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, également appelée un *gestionnaire*, 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

Exemple

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

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

Voir aussi

`List.itemRollOut`

List.labelField

Disponibilité

Flash Player 6 version 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 `"label"`.

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});
```

Voir aussi

`List.labelFunction`

List.labelFunction

Disponibilité

Flash Player 6 version 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;
}
```

Voir aussi

`List.labelField`

List.length

Disponibilité

Flash Player 6 version 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 version 79.

Edition

Flash MX 2004.

Usage

`OccurrenceListe``.maxHPosition`

Description

Propriété : spécifie le nombre de pixels que la liste peut afficher, 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;
```

Voir aussi

`List.hScrollPolicy`

List.multipleSelection

Disponibilité

Flash Player 6 version 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 version 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 version 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 (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*.

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 version 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).

label Chaîne indiquant l'étiquette du nouvel élément.

data 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 version 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 version 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 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.scroll = function(ObjetEvt){
  // 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 : diffusé 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, également appelée un *gestionnaire*, 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 transmet automatiquement un objet événement (*objetEvt*) 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 événement, consultez *Objets événement*, page 592.

Exemple

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

```
form.scroll = function(objEvt){
  trace("liste défilée");
}
maListe.addEventListener("scroll", form);
```

List.selectable

Disponibilité

Flash Player 6 version 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 version 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);
```

Voir aussi

`List.selectedIndices`, `List.selectedItem`, `List.selectedItems`

List.selectedIndices

Disponibilité

Flash Player 6 version 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 ;
```

Voir aussi

`List.selectedIndex`, `List.selectedItem`, `List.selectedItems`

List.selectedItem

Disponibilité

Flash Player 6 version 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);
```

Voir aussi

`List.selectedIndex`, `List.selectedIndices`, `List.selectedItems`

List.selectedItems

Disponibilité

Flash Player 6 version 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;
```

Voir aussi

`List.selectedIndex`, `List.selectedItem`, `List.selectedIndices`

List.setPropertiesAt()

Disponibilité

Flash Player 6 version 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 version 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 `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();
}
```

Voir aussi

`List.sortItemsBy()`

List.sortItemsBy()

Disponibilité

Flash Player 6 version 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");
```

Voir aussi

`List.sortItems()`

List.vPosition

Disponibilité

Flash Player 6 version 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 version 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`.

Voir aussi

`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 27 ou *Classe FocusManager*, page 287.

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.

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.

contentPath URL absolue ou relative indiquant le fichier à charger dans le chargeur. Un chemin relatif doit être relatif au fichier SWF chargeant le contenu. L'URL doit se trouver dans le même sous-domaine que l'URL où se trouve le contenu Flash. Pour une utilisation dans Flash Player ou 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 l'occurrence Logo sur la scène et dans le panneau Inspecteur de composants et 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 programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez la méthode `setSize()` (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.

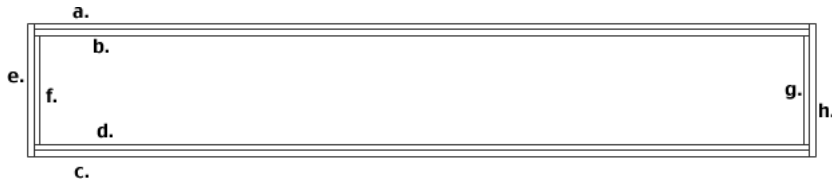
Utilisation d'enveloppes avec le composant Loader

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	Lettre
`borderColor`	a
`highlightColor`	b
`borderColor`	c
`shadowColor`	d
`borderCapColor`	e
`shadowCapColor`	f

Styles RectBorder	Lettre
shadowCapColor	g
borderCapColor	h

Les propriétés de style définissent les positions suivantes sur la bordure :



Classe Loader

Héritage UIObject > UIComponent > View > Loader

Nom de classe ActionScript mx.controls.Loader

Les propriétés de la classe Loader vous permettent de définir le contenu de sorte qu'il se charge et contrôle sa propre progression de chargement 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 27.

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 des classes *UIObject* et *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.

Propriété	Description
`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 des classes *UIObject* et *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 des classes *UIObject* et *UIComponent*.

Loader.autoLoad

Disponibilité

Flash Player 6 version 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 version 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 progression 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(objEvt){
  // objEvt.target est le composant ayant généré l'événement change,
  // c'est-à-dire, 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()`.

Voir aussi

`Loader.bytesTotal`, `UIObject.createClassObject()`

Loader.bytesTotal

Disponibilité

Flash Player 6 version 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 progression 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(objEvt){
  // objEvt.target est le composant ayant généré l'événement change,
  // c'est-à-dire, le Loader.
  pBar.setProgress(loader.bytesLoaded, loader.bytesTotal); // afficher la
  progression
}
loader.addEventListener("progress", loadListener);
loader.content = "logo.swf";
```

Voir aussi

`Loader.bytesLoaded`

Loader.complete

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(complete){
  ...
}
```

Usage 2 :

```
objetDécoute = new Object();
ObjetDécoute.complete = function(objetEvt){
  ...
}
occurrenceDeLoader.addEventListener("complete", ObjetDécoute)
```

Description

Evénement : diffusé 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 (*occurrenceLoader*) distribue un événement (ici, `complete`) qui est géré par une fonction, également appelée un *gestionnaire*, 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 transmet automatiquement un objet événement (*objetEvt*) 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 événement, consultez *Objets événement*, page 592.

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(objEvt){
  loader.visible = true;
}
loader.addEventListener("complete", loadListener);S
loader.contentPath = "logo.swf";
```

Loader.content

Disponibilité

Flash Player 6 version 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.

Voir aussi

`Loader.contentPath`

Loader.contentPath

Disponibilité

Flash Player 6 version 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 dans Flash, tous les fichiers SWF doivent être stockés dans le même dossier et les noms de fichiers ne doivent pas inclure d'informations de dossier ni de disque.

Exemple

L'exemple suivant demande à l'occurrence de chargeur d'afficher le contenu du fichier logo.swf :

```
loader.contentPath = "logo.swf";
```

Loader.load()

Disponibilité

Flash Player 6 version 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 de 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 version 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 de 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(objEvt){
  // objEvt.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 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(progress){
   ...
}
```

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.progress = function(objetEvt){
   ...
}
OccurrenceDeLoader.addEventListener("progress", objetDécoute)
```

Description

Evénement : diffusé 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 (*`occurrenceLoader`*) distribue un événement (ici, `progress`) qui est géré par une fonction, également appelée un *gestionnaire*, 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

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(objEvt){
  // objEvt.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 version 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;
```

Composants de support (Flash Professionnel uniquement)

Les composants de support en flux continu facilitent l'incorporation de supports en flux continu dans les diaporamas Flash. Ces composants vous permettent de présenter vos supports de différentes manières.

Vous pouvez utiliser les trois composants de support suivants :

- Le composant MediaDisplay permet de diffuser un support en flux continu dans votre contenu Flash sans avoir recours à une interface utilisateur. Ce composant peut être utilisé avec des données vidéo et audio. L'utilisateur de votre application ne peut pas contrôler le support lorsque le composant MediaDisplay est utilisé seul.

- Le composant MediaController complète le composant MediaDisplay en fournissant une interface utilisateur qui contrôle la lecture du support à l'aide des contrôles standard (lecture, pause etc.). Les supports ne sont jamais chargés dans le composant MediaController ou lus par ce dernier. Il est uniquement utilisé pour contrôler la lecture dans une occurrence de MediaPlayback ou MediaDisplay. Le composant MediaController comporte un « tiroir » qui affiche le contenu des contrôles de lecture lorsque la souris est positionnée sur le composant.
- Le composant MediaPlayback est une combinaison des composants MediaDisplay et MediaController. Il fournit des méthodes permettant de lire en continu votre contenu multimédia.

N'oubliez pas les points suivants concernant les composants de support :

- Les composants de support requièrent Flash Player 7 ou version ultérieure.
- La fonctionnalité de défilement vers l'avant et l'arrière n'est pas prise en charge par les composants de support. Vous pouvez néanmoins arriver au même résultat en déplaçant le curseur de lecture.
- Seules la taille du composant et la politique du contrôleur sont reflétées dans l'aperçu en direct.
- Les composants de support ne prennent pas en charge pas l'accessibilité.

Interaction avec les composants de support (Flash Professionnel uniquement

Les composants en flux continu MediaPlayback et MediaController répondent aux interactions avec le clavier et la souris. Le composant MediaDisplay ne répond pas aux événements de clavier ou de souris. Le tableau suivant résume les actions des composants MediaPlayback et MediaController lorsqu'ils reçoivent le focus :

Cible	Navigation	Description
Contrôles de lecture d'un contrôleur donné	Survol de la souris	Le bouton est mis en évidence.
Contrôles de lecture d'un contrôleur donné	Clic unique sur le bouton gauche de la souris	Les utilisateurs peuvent contrôler la lecture de support audio et vidéo par le biais des contrôles de lecture d'un contrôleur donné en cliquant sur les contrôles de lecture pour déclencher les effets correspondants. Les boutons Pause/Lire et Rembobiner/Atteindre la fin suivent les comportements standard des boutons. Lorsque l'utilisateur appuie sur le bouton de la souris, le bouton à l'écran apparaît dans son état Enfoncé. Lorsque l'utilisateur relâche le bouton de la souris, le bouton reprend son apparence non sélectionnée. Le bouton Atteindre la fin est désactivé lors de la lecture des fichiers de support FLV.

Cible	Navigation	Description
Contrôles de curseur d'un contrôleur donné	Déplacez le curseur vers l'avant et vers l'arrière	Le curseur de lecture indique la position de l'utilisateur dans le contenu multimédia. La poignée d'affichage se déplace horizontalement (par défaut) pour indiquer la lecture du début (à gauche) jusqu'à la fin (à droite). Le curseur de lecture se déplace de bas en haut lorsque les contrôles sont orientés à la verticale. Au fur et à mesure que la poignée de l'indicateur se déplace de la gauche vers la droite, elle met en évidence l'espace d'affichage précédent pour indiquer que ce contenu a été lu ou sélectionné. L'espace d'affichage devant la poignée de l'indicateur n'est pas mis en évidence jusqu'au passage de l'indicateur. Les utilisateurs peuvent faire glisser la poignée de l'indicateur pour modifier la position de lecture du contenu multimédia. Si le support est en cours de lecture, la lecture commence automatiquement à l'endroit où l'utilisateur relâche le bouton de la souris. Si le support est en pause, le curseur peut être déplacé puis relâché et le support reste en pause. Vous pouvez également utiliser un curseur de volume, qui peut être déplacé de gauche (muet) à droite (volume maximum) dans les dispositions horizontale et verticale.
Navigation du contrôleur de lecture	Tab, Maj+Tab	Déplace le focus de bouton en bouton dans le composant contrôleur, où l'élément ayant le focus est mis en évidence. Cette navigation fonctionne avec les contrôles Pause/Lire, Rembobiner, Atteindre la fin, et les contrôles de volume désactivé et de volume max. Le focus se déplace de gauche à droite et de haut en bas lorsque les utilisateurs passent d'un élément à l'autre. L'utilisation de Maj+Tab déplace le focus de droite à gauche et de bas en haut. Lorsqu'il reçoit le focus via la touche Tab, le contrôle transmet immédiatement le focus au bouton Lire/Pause. Lorsque le focus est sur le bouton de volume maximum puis que la touche Tab est enfoncée, le contrôle donne le focus au contrôle suivant dans l'ordre des tabulations sur la scène.
Bouton de contrôle donné	Espace ou Entrée/ Retour	Sélectionne l'élément ayant le focus. Lorsqu'il est enfoncé, le bouton apparaît dans son état Enfoncé. Lorsqu'il est relâché, le bouton reprend son état Survolé avec focus.

Présentation des composants de support (Flash Professionnel uniquement)

Avant de commencer à utiliser des composants de support, il est conseillé de se familiariser avec leur mode de fonctionnement. Cette section fournit un aperçu du mode de fonctionnement des composants de support. La plupart des propriétés répertoriées dans cette section peuvent être directement définies à l'aide du panneau Inspecteur de composants. Pour plus d'informations, consultez *Utilisation du panneau Inspecteur de composants avec les composants de support*, page 352.

Les propriétés suivantes peuvent être définies pour les composants MediaDisplay et MediaPlayback, à l'exception des propriétés de disposition, qui sont présentées plus loin dans cette section :

- Le type de support, qui peut être défini sur MP3 ou FLV (consultez `Media.mediaType` et `Media.setMedia()`).
- Le chemin relatif ou absolu, qui contient le fichier de support à lire en continu (consultez `Media.contentPath`).
- Les objets point de repère, ainsi que leur nom, heure et propriétés de lecteur (consultez `Media.addCuePoint()` et `Media.cuePoints`). Le nom du point de repère est aléatoire et doit être défini de manière à avoir un sens lors de l'utilisation d'événements d'écoute et de trace. Un point de repère diffuse un événement `cuePoint` lorsque la valeur de sa propriété time est égale à celle de l'emplacement de la tête de lecture du composant MediaPlayback ou MediaDisplay auquel il est associé. La propriété player est une référence à l'occurrence de MediaPlayback à laquelle elle est associée. Les points de repère peuvent ensuite être supprimés par le biais de `Media.removeCuePoint()` et `Media.removeAllCuePoints()`.

Les composants de support en flux continu diffusent plusieurs événements associés. Les événements de diffusion suivants peuvent être utilisés pour définir d'autres éléments en mouvement :

- Un événement `change` est diffusé en continu par les composants MediaDisplay et MediaPlayback pendant la lecture du support. Pour plus d'informations, consultez `Media.change`.
- Un événement `progress` est diffusé en continu par les composants MediaDisplay et MediaPlayback pendant le chargement du support. Pour plus d'informations, consultez `Media.progress`.
- Un événement `click` est diffusé par les composants MediaController et MediaPlayback chaque fois qu'un utilisateur clique sur le bouton Pause/Lire. Dans ce cas, la propriété `detail` de l'objet événement fournit des informations sur le bouton qui a été cliqué. Pour plus d'informations, consultez `Media.click`.
- Un événement `volume` est diffusé par les composants MediaController et MediaPlayback lorsque les contrôles de volume sont réglés par l'utilisateur. Pour plus d'informations, consultez `Media.volume`.
- Un événement `playheadChange` est diffusé par les composants MediaController et MediaPlayback lorsque le curseur de lecture est déplacé par l'utilisateur. Pour plus d'informations, consultez `Media.playheadChange`.

Le composant MediaDisplay fonctionne en parallèle avec le composant MediaController. Lorsqu'ils sont combinés, les composants se comportent d'une manière similaire à celle du composant MediaPlayback. Ils permettent néanmoins une plus grande flexibilité par rapport à la disposition. Ainsi, si vous recherchez un aspect flexible pour la présentation de votre support, utilisez les composants MediaDisplay et MediaController. Sinon, le composant MediaPlayback est le meilleur choix.

Présentation du composant MediaDisplay

Lorsque vous placez un composant MediaDisplay sur la scène, il apparaît sans interface utilisateur visible. Il s'agit simplement d'un conteneur destiné à contenir et lire des supports. L'apparence de tout support vidéo lu dans un composant MediaDisplay est affectée par les propriétés suivantes :

- `Media.aspectRatio`
- `Media.autoSize`
- Hauteur
- Largeur

Remarque : L'utilisateur ne pourra rien voir, excepté si un support est en cours de lecture.

La propriété `Media.aspectRatio` est prioritaire par rapport aux autres propriétés. Lorsque `Media.aspectRatio` est défini sur `true` (par défaut), le composant réajuste systématiquement la taille du support en cours de lecture une fois que la taille du composant a été définie, pour s'assurer que les proportions du support sont conservées.

Pour les fichiers FLV, lorsque `Media.autoSize` est défini sur `true`, le support à lire est affiché à sa taille préférée, sans tenir compte de la taille du composant. Ceci implique que, à moins que la taille de l'occurrence MediaDisplay ne soit la même que celle du support, le support dépassera les limites de l'occurrence ou ne remplira pas la taille de l'occurrence. Lorsque `Media.autoSize` est défini sur `false`, la taille de l'occurrence est respectée autant que possible, tout en tenant compte des proportions. Si `Media.autoSize` et `Media.aspectRatio` sont tous deux définis sur `false`, la taille exacte du composant est utilisée.

Remarque : Etant donné qu'aucune image n'est affichée pour les fichiers MP3, il est inutile de définir `Media.autoSize`, car cela n'aura aucun effet. Pour les fichiers MP3, la taille minimum utilisable est de 60 pixels de hauteur sur 256 pixels de largeur dans l'orientation par défaut.

Le composant MediaDisplay prend également en charge la propriété `Media.volume`. Cette propriété prend des valeurs entières de 0 à 100, 0 correspondant à muet et 100 correspondant au volume maximum. Le paramètre par défaut est 75.

Présentation du composant MediaController

L'interface pour le composant MediaController dépend de ses propriétés `Media.controllerPolicy` et `Media.backgroundStyle`. La propriété `Media.controllerPolicy` détermine si le jeu de contrôles du support est toujours visible, réduit ou uniquement visible lorsque la souris passe sur la zone de contrôle du composant. Lorsqu'il est réduit, le contrôleur affiche une barre de progression modifiée, qui est une combinaison de la barre de chargement et de la barre de lecture. Elle affiche la progression des octets chargés en bas de la barre, et la progression de la tête de lecture est affichée juste au dessus. L'état développé affiche une version étendue de la barre de lecture/chargement, qui contient les éléments suivants :

- Des étiquettes de texte sur la gauche indiquant l'état de la lecture (en flux continu ou en pause), et sur la droite l'emplacement de la tête de lecture en secondes.
- Un indicateur d'emplacement de la tête de lecture.
- Un curseur que les utilisateurs peuvent faire glisser pour naviguer dans le support.

Les éléments suivants sont également fournis avec le composant MediaController :

- Un bouton d'état Lire/Pause.

- Un groupe de deux boutons : Rembobiner et Atteindre la fin, qui naviguent respectivement vers le début et vers la fin du support.
- Un contrôle de volume qui comprend un curseur, un bouton muet et un bouton volume maximum.

Les états réduit et développé du composant MediaController utilisent tous deux la propriété `Media.backgroundStyle`. Cette propriété indique si le contrôleur dessine un arrière-plan chromé (par défaut) ou s'il autorise l'affichage de l'arrière-plan de l'animation depuis les contrôles.

Le composant MediaController possède un paramètre d'orientation, `Media.horizontal`, qui peut être utilisé pour dessiner le composant avec une orientation horizontale (par défaut) ou verticale. Dans une orientation horizontale, la barre de lecture suit le support en cours de lecture de la gauche vers la droite. Dans une orientation verticale, la barre de lecture suit le support de bas en haut.

Les composants MediaDisplay et MediaController peuvent être associés l'un à l'autre via les méthodes `Media.associateDisplay()` et `Media.associateController()`. Lorsqu'elles sont appelées, ces méthodes permettent à l'occurrence MediaController de mettre à jour ses contrôles en fonction des événements diffusés depuis l'occurrence MediaDisplay. Elles permettent également au composant MediaDisplay de réagir aux paramètres définis par l'utilisateur depuis le composant MediaController.

Présentation du composant MediaPlayback

Le composant MediaPlayback est une combinaison des contrôles de MediaController et de MediaDisplay. Les deux sous-composants sont contenus dans MediaPlayback. Les portions de MediaController et de MediaDisplay sont toujours redimensionnées pour correspondre à la taille de l'occurrence générale du composant MediaPlayback.

Le composant MediaPlayback utilise `Media.controlPlacement` pour définir la disposition des contrôles. Les emplacements de contrôle pouvant être utilisés sont `haut`, `bas`, `gauche` et `droite`, indiquant l'endroit où les contrôles seront affichés par rapport à l'affichage. Par exemple, la valeur `droite` donne une orientation verticale au contrôle et le positionne sur la droite de l'affichage.

Utilisation des composants de support (Flash Professionnel uniquement)

L'utilisation croissante de supports pour fournir des informations aux utilisateurs Web implique de fournir à ces mêmes utilisateurs un moyen de lire ces supports en flux continu et de les contrôler. Les exemples suivants sont des scénarios d'utilisation pour les composants de support :

- Affichage d'un support présentant une société
- Lecture en flux continu d'animations ou d'aperçus d'animations
- Lecture en continu de chansons ou de fragments de chansons
- Fourniture de matériel de formation par le biais de supports

Utilisation du composant MediaPlayback

Admettons que vous devez développer un site Web pour vos clients qui permet aux utilisateurs du site de prévisualiser des DVD et CD que vous vendez dans un environnement de supports enrichis. L'exemple ci-dessous montre les étapes à suivre pour y parvenir et part du principe que votre site Web est prêt à recevoir des composants en flux continu.

Pour créer un document Flash qui affiche un aperçu de CD ou de DVD :

1 Dans Flash, choisissez Fichier > Nouveau, puis sélectionnez Document Flash.

2 Ouvrez le panneau Composants (Fenêtre > Panneaux de développement > Composants) et double-cliquez sur le composant MediaPlayback, pour placer une occurrence du composant sur la scène.

3 Sélectionnez l'occurrence du composant MediaPlayback et entrez le nom d'occurrence **monSupport** dans l'inspecteur des propriétés.

4 Dans le panneau Inspecteur de composants (Fenêtre > Panneaux de développement > Inspecteur de composants), définissez votre type de support en fonction du type de support qui sera lu en flux continu (MP3 ou FLV).

5 Si vous avez sélectionné FLV, entrez la durée de la vidéo dans les champs de texte Video Length. Utilisez le format HH:MM:SS.

6 Entrez l'emplacement de votre aperçu de vidéo dans le champ de texte URL. Vous pouvez par exemple entrer **http://mon.web.com/aperçusvideo/NomDAnimation.flv**.

7 Définissez les options souhaitées pour les cases à cocher Automatically Play, Use Preferred Media Size et Respect Aspect Ratio.

8 Définissez l'emplacement du contrôle du côté désiré du composant MediaPlayback.

9 Ajoutez un point de repère vers la fin du support. Il sera utilisé avec un écouteur pour ouvrir une fenêtre contextuelle informant l'utilisateur que l'animation est en vente. Nommez le point de repère **nomPointRepère** et définissez son heure pour qu'il se trouve vers la fin du clip (à quelques secondes de la fin). Pour ce faire, suivez les étapes ci-après :

   a Double-cliquez sur un composant Window pour le faire apparaître sur la scène.

   b Supprimez le composant Window. Ceci a pour effet de placer un élément nommé Window dans votre bibliothèque.

   c Créez un champ de texte et entrez un texte informant l'utilisateur que l'animation est à vendre.

   d Convertissez ce champ de texte en un clip en choisissant Modifier > Convertir en symbole, et nommez-le **maVente_mc**.

   e Cliquez avec le bouton droit de la souris sur le clip `maVente_mc` dans la bibliothèque, sélectionnez Liaison et activez l'option Exporter pour ActionScript. Un clip est placé dans la bibliothèque d'exécution.

10 Ajoutez le code ActionScript suivant à l'image 1. Ce code crée un écouteur qui ouvre une fenêtre contextuelle informant l'utilisateur que l'animation est en vente.

```
// Importe les classes nécessaires pour créer dynamiquement la fenêtre
  contextuelle

import mx.containers.Window;
import mx.managers.PopUpManager;

// Créez un objet d'écoute pour lancer la fenêtre contextuelle de vente
var écouteurVente = new Object();

écouteurVente.cuePoint = function(evt){

var venteGagnée = PopUpManager.createPopUp(_root, Window, false,
  {closeButton: true, title: "Vente animation ", contentPath:
  "maVente_mc"});

// Agrandissez la fenêtre de sorte que le contenu puisse être inclus
```

```
venteGagnée.setSize(80, 80);
var supprVenteGagnée = new Object();
supprVenteGagnée.click = function(evt){
venteGagnée.deletePopUp();
}
venteGagnée.addEventListener("click", supprVenteGagnée);

}

monSupport.addEventListener("cuePoint", écouteurVente);
```

Utilisation des composants MediaDisplay et MediaController

Supposons que vous décidiez d'avoir plus de contrôle sur l'aspect de l'affichage de votre support. Vous devez utiliser conjointement les composants MediaDisplay et MediaController pour atteindre le résultat escompté. L'exemple suivant présente les étapes équivalentes de l'exemple précédent permettant de créer une application Flash qui affiche le support d'aperçu de votre CD ou DVD.

Pour créer un document Flash qui affiche un aperçu de CD ou de DVD :

1. Dans Flash, choisissez Fichier > Nouveau, puis sélectionnez Document Flash.
2. Dans le panneau Composants (Fenêtre > Panneaux de développement > Composants), double-cliquez sur les composants MediaController et MediaDisplay. Une occurrence de chaque composant est alors placée sur la scène.
3. Sélectionnez l'occurrence de MediaDisplay et entrez **monAffichage** comme nom d'occurrence dans l'inspecteur des propriétés.
4. Sélectionnez l'occurrence de MediaController et entrez **monContrôleur** comme nom d'occurrence dans l'inspecteur des propriétés.
5. Ouvrez le panneau Inspecteur de composants depuis l'inspecteur des propriétés et définissez le type de votre support en fonction du support qui sera diffusé en flux continu (MP3 ou FLV).
6. Si vous avez sélectionné FLV, entrez la durée de la vidéo dans les champs de texte Video Length au format HH:MM:SS.
7. Entrez l'emplacement de votre aperçu de vidéo dans le champ de texte URL. Vous pouvez par exemple entrer **http://mon.web.com/aperçusvideo/NomDAnimation.flv**.
8. Définissez les options souhaitées pour les cases à cocher Automatically Play, Use Preferred Media Size et Respect Aspect Ratio.
9. Sélectionnez l'occurrence de MediaController et, dans le panneau Inspecteur de composants, choisissez une orientation verticale en définissant la propriété `horizontal` sur `false`.
10. Dans le panneau Inspecteur de composants, définissez `backgroundStyle` sur None.

    Ceci spécifie que l'occurrence de MediaController ne doit pas dessiner d'arrière-plan mais remplir le support entre les contrôles.
11. Utilisez un comportement pour associer les occurrences de MediaController et de MediaDisplay de sorte que l'occurrence de MediaController reflète précisément les déplacements de la tête de lecture ainsi que d'autres paramètres de l'occurrence de MediaDisplay, et que l'occurrence de MediaDisplay réponde aux clics de l'utilisateur :
    a. Sélectionnez l'occurrence de MediaDisplay et, dans l'inspecteur des propriétés, entrez **monAffichageDeSupport** comme nom d'occurrence.
    b. Sélectionnez l'occurrence de MediaController qui déclenche le comportement.

c Dans le panneau Comportements (Fenêtre > Panneaux de développement > Comportements), cliquez sur le bouton Ajouter un comportement (+) et sélectionnez Médias > Affichage associé.

d Dans la fenêtre Affichage associé, sélectionnez `monAffichageDeSupport` sous `_root` et cliquez sur OK.

Pour plus d'informations sur l'utilisation des comportements avec les composants de support, consultez *Contrôle des composants de support à l'aide des comportements*, page 353.

Utilisation du panneau Inspecteur de composants avec les composants de support

Le panneau Inspecteur de composants facilite notamment la définition des paramètres et des propriétés des composants de support. Pour utiliser ce panneau, cliquez sur le composant souhaité sur la scène et, l'inspecteur des propriétés étant ouvert, cliquez sur Ouvrir l'Inspecteur de composants. Le panneau Inspecteur de composants peut être utilisé de différentes manières :

- Pour lire le support automatiquement (consultez `Media.activePlayControl` et `Media.autoPlay`)
- Pour conserver ou ignorer les proportions du support (consultez `Media.aspectRatio`)
- Pour déterminer si le support est automatiquement redimensionné en fonction de la taille de l'occurrence du composant (consultez `Media.autoSize`)
- Pour activer ou désactiver l'arrière-plan chromé (consultez `Media.backgroundStyle`)
- Pour spécifier le chemin de votre support sous la forme d'une URL (consultez `Media.contentPath`)
- Pour spécifier la visibilité des contrôles de lecture (consultez `Media.controllerPolicy`)
- Pour ajouter des objets point de repère (consultez `Media.addCuePoint()`)
- Pour supprimer des objets point de repère (consultez `Media.removeCuePoint()`)
- Pour définir l'orientation des occurrences de MediaController (consultez `Media.horizontal`)
- Pour définir le type du support lu (consultez `Media.setMedia()`)
- Pour définir la durée de lecture du support FLV (consultez `Media.totalTime`)
- Pour définir les derniers chiffres du temps affiché en indiquant des millisecondes ou des images par seconde (ips)

Il est important de se familiariser avec certains concepts lorsque vous utilisez le panneau Inspecteur de composants :

- Le contrôle de durée de la vidéo est supprimé lorsque le type de vidéo MP3 est sélectionné, car les informations sont lues automatiquement lorsque des fichiers MP3 sont utilisés. Pour les fichiers FLV, vous devez ajouter le temps total du support (`Media.totalTime`) pour que la barre de lecture du composant MediaPlayback (ou de tout composant MediaController d'écoute) puisse refléter précisément la progression de la lecture.

- Si vous définissez le type de fichier sur FLV, une option Milliseconds s'affiche et (si l'option Milliseconds est désactivée) un menu contextuel FPS apparaît. Lorsque l'option Milliseconds est activée, le contrôle FPS est masqué. Dans ce mode, la durée affichée dans la barre de lecture lors de l'exécution est au format HH:MM:SS.mmm (H = heures, M = minutes, S = secondes, m = millisecondes) et les points de repère sont définis en secondes. Lorsque l'option Milliseconds est désactivée, le contrôle FPS est activé et la durée de la barre de lecture est au format HH:MM:SS.II (I = images par seconde), alors que les points de repère sont définis par image.

  Remarque : Vous pouvez uniquement définir la propriété FPS en utilisant le panneau Inspecteur de composants. La définition d'une valeur fps à l'aide d'ActionScript n'a aucun effet et sera ignorée.

Contrôle des composants de support à l'aide des comportements

Les comportements sont des scripts ActionScript prêts à l'emploi ajoutés à une occurrence d'objet, tel qu'un composant MediaDisplay, pour le contrôler. Les comportements vous permettent d'ajouter la puissance, le contrôle et la flexibilité du codage ActionScript à votre document sans avoir à créer le code ActionScript vous-même.

Pour contrôler un composant de support à l'aide d'un comportement, vous devez utiliser le panneau Comportements afin d'appliquer le comportement à une occurrence de composant de support donnée. Définissez l'événement déclencheur du comportement (par exemple le fait d'atteindre un point de repère spécifique), sélectionnez un objet cible (les composants de support qui seront affectés par le comportement) et, si nécessaire, sélectionnez les paramètres du comportement (par exemple le clip dans le support vers lequel naviguer).

Le tableau ci-dessous présente les comportements disponibles dans Flash MX 2004 Professionnel, permettant de contrôler les composants de support intégrés.

Comportement	Objectif	Paramètres
Contrôleur associé	Associe un composant MediaController à un composant MediaDisplay	Nom d'occurrence des composants MediaController cibles
Affichage associé	Associe un composant MediaDisplay à un composant MediaController	Nom d'occurrence des composants MediaController cibles
Exploration des points de repère de l'image nommée	Place une action sur une occurrence de MediaDisplay ou de MediaPlayback qui indique à un clip spécifié de naviguer vers une image portant le même nom qu'un point de repère donné	Nom de l'image et nom du point de repère (les noms doivent être identiques)
Exploration des points de repère de la diapositive	Fait naviguer un document Flash basé sur des diapositives jusqu'à une diapositive portant le même nom qu'un point de repère donné	Nom de la diapositive et nom du point de repère (les noms doivent être identiques)

Pour associer un composant MediaDisplay à un composant MediaController :

1 Placez une occurrence de MediaDisplay et une occurrence de MediaController sur la scène.

2 Sélectionnez l'occurrence de MediaDisplay et, dans l'inspecteur des propriétés, entrez **monAffichageDeSupport** comme nom d'occurrence.

3 Sélectionnez l'occurrence de MediaController qui déclenche le comportement.

4 Dans le panneau Comportements (Fenêtre > Panneaux de développement > Comportements), cliquez sur le bouton Ajouter un comportement (+) et sélectionnez Médias > Affichage associé.

5 Dans la fenêtre Affichage associé, sélectionnez `monAffichageDeSupport` sous `_root` et cliquez sur OK.

Remarque : Si vous avez associé le composant MediaDisplay au composant MediaController, il n'est pas nécessaire d'associer le composant MediaController au composant MediaDisplay.

Pour associer un composant MediaController à un composant MediaDisplay :

1 Placez une occurrence de MediaDisplay et une occurrence de MediaController sur la scène.

2 Sélectionnez l'occurrence de MediaController et, dans l'inspecteur des propriétés, entrez **monContrôleurDeSupport** comme nom d'occurrence.

3 Sélectionnez l'occurrence de MediaDisplay qui déclenche le comportement.

4 Dans le panneau Comportements (Fenêtre > Panneaux de développement > Comportements), cliquez sur le bouton Ajouter un comportement (+) et sélectionnez Médias > Contrôleur associé.

5 Dans la fenêtre Contrôleur associé, sélectionnez `monContrôleurDeSupport` sous `_root` et cliquez sur OK.

Pour utiliser un comportement Exploration des points de repère de l'image nommée :

1 Placez une occurrence du composant MediaDisplay ou MediaPlayback sur la scène.

2 Sélectionnez l'image vers laquelle vous souhaitez que le support navigue et, à l'aide de l'inspecteur des propriétés, entrez **monImageNommée** comme nom d'image.

3 Sélectionnez votre occurrence de MediaDisplay ou de MediaPlayback.

4 Dans le panneau Inspecteur de composants, cliquez sur le bouton Ajouter (+), entrez l'heure du point de repère au format HH:MM:SS:mmm ou HH:MM:SS:II, et nommez le point de repère **monImageNommée**.

Le point de repère indique la durée devant s'écouler avant que vous ne naviguiez vers l'image sélectionnée. Par exemple, si vous souhaitez atteindre `monImageNommée` 5 secondes après le début de l'animation, entrez **5** dans le champ de texte SS et entrez **monImageNommée** dans le champ de texte Name.

5 Dans le panneau Comportements (Fenêtre > Panneaux de développement > Comportements), cliquez sur le bouton Ajouter un comportement (+) et sélectionnez Médias > Exploration des points de repère de l'image nommée.

6 Dans la fenêtre Exploration des points de repère de l'image nommée, sélectionnez le clip `_root` et cliquez sur OK.

Pour utiliser un comportement Exploration des points de repère d'une diapositive :

1 Ouvrez votre nouveau document comme un diaporama Flash.

2 Placez une occurrence du composant MediaDisplay ou MediaPlayback sur la scène.

3 Dans le panneau situé à gauche de la scène, cliquez sur le bouton Insérer un écran (+) pour ajouter une deuxième diapositive, puis sélectionnez la deuxième diapositive et renommez-la **maDiapositive**.

4 Sélectionnez votre occurrence de MediaDisplay ou de MediaController.

5 Dans le panneau Inspecteur de composants, cliquez sur le bouton Ajouter (+) et entrez l'heure du point de repère au format HH:MM:SS:mmm ou HH:MM:SS:II, et nommez le point de repère **maDiapositive**.

Le point de repère indique la durée devant s'écouler avant que vous ne naviguiez vers la diapositive sélectionnée. Par exemple, si vous souhaitez atteindre `maDiapositive` 5 secondes après le début de l'animation, entrez 5 dans le champ de texte SS et entrez **maDiapositive** dans le champ de texte Name.

6 Dans le panneau Comportements (Fenêtre > Panneaux de développement > Comportements), cliquez sur le bouton Ajouter un comportement (+) et sélectionnez Médias > Exploration des points de repère d'une diapositive.

7 Dans la fenêtre Exploration des points de repère d'une diapositive, sélectionnez `Présentation` sous le clip `_root` et cliquez sur OK.

Paramètres des composants de support (Flash Professionnel uniquement)

Les tableaux suivants répertorient les paramètres de programmation que vous pouvez définir pour une occurrence de composant de support donnée dans l'inspecteur des propriétés :

Paramètres du composant MediaDisplay

Nom	Type	Valeur par défaut	Description
Automatically Play (`Media.autoPlay`)	Booléen	Sélectionné	Indique si le support est lu dès la fin de son chargement.
Use Preferred Media Size (`Media.autoSize`)	Booléen	Sélectionné	Indique si le support associé à l'occurrence de MediaDisplay correspond à la taille du composant ou utilise simplement sa taille par défaut.
FPS	Entier	30	Indique le nombre d'images par seconde. Lorsque l'option Milliseconds est activée, ce contrôle est désactivé.
Points de repère (`Media.cuePoints`)	Tableau	Non définis	Tableau d'objets point de repère, chaque objet possédant un nom et une position dans le temps dans un format HH:MM:SS:II (option Milliseconds activée) ou HH:MM:SS:mmm valide.
FLV ou MP3 (`Media.mediaType`)	"FLV" ou "MP3" ()	"FLV"	Indique le type de support à lire.
Milliseconds	Booléen	Non sélectionné	Indique si la barre de lecture utilise des images ou des millisecondes et si les points de repère utilisent des secondes ou des images. Lorsque cette option est activée, le contrôle FPS est masqué.
URL (`Media.contentPath`)	Chaîne	Non défini	Chaîne contenant le chemin et le nom de fichier du support à lire.
Video Length (`Media.totalTime`)	Entier	Non défini	Le temps total nécessaire pour lire le support FLV. Ce paramètre est requis pour que la barre de lecture fonctionne correctement. Ce contrôle est uniquement visible lorsque le type de support est défini sur FLV.

Paramètres du composant MediaController

Nom	Type	Valeur par défaut	Description
activePlayControl (`Media.activePlayControl`)	Chaîne : "pause" ou "play"	"pause"	Détermine si la barre de lecture est en mode pause ou play au moment de l'instanciation.
backgroundStyle (`Media.backgroundStyle`)	Chaîne : "default" ou "none"	"default"	Indique si l'arrière-plan chromé doit être dessiné pour l'occurrence de MediaController.
controllerPolicy (`Media.controllerPolicy`)	"auto", "on" ou "off"	"auto"	Détermine si le contrôleur s'ouvre ou se ferme en fonction de la position de la souris, ou est verrouillé dans l'état ouvert ou fermé.
horizontal (`Media.horizontal`)	Booléen	true	Détermine si la portion de contrôleur de l'occurrence est orientée verticalement ou horizontalement. Une valeur `true` indique que le composant aura une position horizontale.
enabled	Booléen	true	Détermine si ce contrôle peut être modifié par l'utilisateur. Une valeur `true` indique que le contrôle peut être modifié.
visible	Booléen	true	Détermine si ce contrôle peut être affiché par l'utilisateur. Une valeur `true` indique que le contrôle peut être affiché.
minHeight	Entier	0	Hauteur minimale autorisée pour cette occurrence, en pixels.
minWidth	Entier	0	Largeur minimale autorisée pour cette occurrence, en pixels.

Paramètres du composant MediaPlayback

Nom	Type	Valeur par défaut	Description
Control Placement (`Media.controlPlacement`)	"top", "bottom", "left" et "right"	"bottom"	Position du contrôleur. La valeur est liée à l'orientation.
`Media.controllerPolicy`	Booléen	true	Détermine si le contrôleur s'ouvre ou se ferme en fonction de la position de la souris.
Automatically Play (`Media.autoPlay`)	Booléen	Sélectionné	Indique si le support est lu dès la fin de son chargement.

Nom	Type	Valeur par défaut	Description
Use Preferred Media Size (`Media.autoSize`)	Booléen	Sélectionné	Détermine si l'occurrence de MediaController est redimensionnée en fonction de la taille du support ou si elle utilise d'autres paramètres.
FPS	Entier	30	Nombre d'images par seconde. Lorsque l'option Milliseconds est activée, ce contrôle est désactivé.
Points de repère (`Media.cuePoints`)	Tableau	Non définis	Tableau d'objets point de repère, chaque objet possédant un nom et une position dans le temps dans un format HH:MM:SS:mmm (option Milliseconds activée) ou HH:MM:SS:Il valide.
FLV ou MP3 (`Media.mediaType`)	"FLV" ou "MP3" ()	"FLV"	Indique le type de support à lire.
Milliseconds	Booléen	Non sélectionné	Indique si la barre de lecture utilise des images ou des millisecondes et si les points de repère utilisent des secondes ou des images. Lorsque cette option est activée, le contrôle FPS est désactivé.
URL (`Media.contentPath`)	Chaîne	Non défini	Chaîne contenant le chemin et le nom de fichier du support à lire.
Video Length (`Media.totalTime`)	Entier	Non défini	Le temps total nécessaire pour lire le support FLV. Ce paramètre est requis pour que la barre de lecture fonctionne correctement.

Création d'applications avec les composants de support (Flash Professionnel uniquement)

Il est relativement facile de créer du contenu Flash à l'aide des composants de support, et ce en quelques étapes seulement.

Cet exemple montre comment créer une application pour lire un petit fichier de support accessible au public.

Pour ajouter un composant de support à une application :

1. Dans Flash, choisissez Fichier > Nouveau, puis sélectionnez Document Flash.
2. Dans le panneau Composants (Fenêtre > Panneaux de développement > Composants), double-cliquez sur le composant MediaPlayback pour l'ajouter sur la scène.
3. Dans l'inspecteur des propriétés, entrez **monSupport** comme nom d'occurrence.
4. Dans l'Inspecteur des propriétés, cliquez sur Ouvrir l'Inspecteur de composants.
5. Dans le panneau Inspecteur de composants, entrez **http://www.cathphoto.com/c.flv** dans le champ de texte URL.
6. Choisissez Contrôle > Tester l'animation pour visualiser la lecture du support.

Personnalisation des composants de support (Flash Professionnel uniquement)

Vous pouvez appliquer des enveloppes pour modifier l'apparence de vos composants de support. Pour obtenir un guide complet sur la personnalisation des composants, consultez le Chapitre 3, *Personnalisation des composants*, page 29.

Utilisation des styles avec les composants de support

Les styles ne sont pas supportés avec les composants de support.

Utilisation d'enveloppes avec les composants de support

Les composants de support ne supportent pas le réhabillage dynamique, même si vous pouvez ouvrir le document source des composants de support et modifier leurs actifs pour obtenir l'aspect escompté. Il est préférable de faire une copie de ce fichier et d'utiliser cette copie, de façon à pouvoir toujours revenir au document source installé. Le document source des composants de support se trouve aux emplacements suivants :

- Windows : C:\Documents and Settings*utilisateur*\Local Settings\Application Data\Macromedia\Flash MX 2004*langue*\Configuration\ComponentFLA.fla
- Macintosh : Disque dur:Users:*nomUtilisateur*:Library:Application Support:Macromedia:Flash MX 2004:*langue*:Configuration:ComponentFLA.fla

Pour plus d'informations sur les enveloppes de composants, consultez *A propos de l'application des enveloppes aux composants*, page 38.

Classe Media (Flash Professionnel uniquement)

Héritage mx.core.UIComponent

Noms de classe ActionScript mx.controls.MediaController, mx.controls.MediaDisplay, mx.controls.MediaPlayback

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

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

Remarque : Le code `trace(monOccurrenceDeSupport.version);` renvoie `undefined`.

Méthodes de la classe Media

Méthode	Composants	Description
`Media.addCuePoint()`	MediaDisplay, MediaPlayback	Ajoute un objet point de repère à l'occurrence du composant.
`Media.associateController()`	MediaDisplay	Associe une occurrence de MediaDisplay à une occurrence de MediaController.
`Media.associateDisplay()`	MediaController	Associe une occurrence de MediaController à une occurrence de MediaDisplay.

Méthode	Composants	Description
Media.displayFull()	MediaPlayback	Convertit l'occurrence du composant en mode de lecture plein écran.
Media.displayNormal()	MediaPlayback	Reconvertit l'occurrence de composant à sa taille d'écran originale.
Media.getCuePoint()	MediaDisplay, MediaPlayback	Renvoie un objet point de repère.
Media.play()	MediaDisplay, MediaPlayback	Lit le support associé à l'occurrence de composant à un point de départ donné.
Media.pause()	MediaDisplay, MediaPlayback	Met la tête de lecture en pause à son emplacement actuel dans le scénario du support.
Media.removeAllCuePoints()	MediaDisplay, MediaPlayback	Supprime tous les objets point de repère associés à une occurrence de composant donnée.
Media.removeCuePoint()	MediaDisplay, MediaPlayback	Supprime un point de repère spécifié associé à une occurrence de composant donnée.
Media.setMedia()	MediaDisplay, MediaPlayback	Définit le type et le chemin du support sur le type de support spécifié.
Media.stop()	MediaDisplay, MediaPlayback	Arrête la tête de lecture et la déplace à la position 0, qui correspond au début du support.

Propriétés de la classe Media

Propriété	Composants	Description
Media.activePlayControl	MediaController	Détermine l'état du composant lors de son chargement à l'exécution.
Media.aspectRatio	MediaDisplay, MediaPlayback	Détermine si l'occurrence du composant conserve les proportions de sa vidéo.
Media.autoPlay	MediaDisplay, MediaPlayback	Détermine si l'occurrence du composant est immédiatement mise en mémoire tampon et lue.
Media.autoSize	MediaDisplay, MediaPlayback	Détermine comment la portion d'affichage de support du composant MediaDisplay ou MediaPlayback est redimensionnée.
Media.backgroundStyle	MediaController	Détermine si l'occurrence du composant dessine son arrière-plan chromé.
Media.bytesLoaded	MediaDisplay, MediaPlayback	Nombre d'octets chargés pouvant être lus.
Media.bytesTotal	MediaDisplay, MediaPlayback	Nombre d'octets à charger dans l'occurrence du composant.
Media.contentPath	MediaDisplay, MediaPlayback	Chaîne contenant le chemin relatif et le nom de fichier du support devant être diffusé en flux continu et lu.

Propriété	Composants	Description
Media.controllerPolicy	MediaController, MediaPlayback	Détermine si les contrôles contenus dans le composant sont masqués pendant la lecture et uniquement affichés lorsqu'un événement de survol est déclenché ou lorsque les contrôles sont visibles ou masqués en permanence.
Media.controlPlacement	MediaPlayback	Détermine si les contrôles du composant sont positionnés par rapport au composant.
Media.cuePoints	MediaDisplay, MediaPlayback	Tableau d'objets point de repère ayant été associés à une occurrence de composant donnée.
Media.horizontal	MediaController	Détermine l'orientation de l'occurrence du composant.
Media.mediaType	MediaDisplay, MediaPlayback	Détermine le type de support à lire.
Media.playheadTime	MediaDisplay, MediaPlayback	Contient la position actuelle de la tête de lecture (en secondes) pour le scénario de support en cours de lecture.
Media.playing	MediaDisplay, MediaPlayback	Renvoie une valeur booléenne indiquant si une occurrence de composant donnée lit un support.
Media.preferredHeight	MediaDisplay, MediaPlayback	Valeur par défaut de la hauteur d'un fichier de support FLV.
Media.preferredWidth	MediaDisplay, MediaPlayback	Valeur par défaut de la largeur d'un fichier de support FLV.
Media.totalTime	MediaDisplay, MediaPlayback	Entier indiquant la durée totale du support, en secondes.
Media.volume	MediaDisplay, MediaPlayback	Entier de 0 (minimum) à 100 (maximum) représentant le niveau de volume.

Evénements de la classe Media

Evénement	Composants	Description
Media.change	MediaDisplay, MediaPlayback	Diffusé en continu pendant la lecture du support.
Media.click	MediaController, MediaPlayback	Diffusé lorsque l'utilisateur clique sur le bouton de lecture/pause.
Media.complete	MediaDisplay, MediaPlayback	Notification indiquant que la tête de lecture a atteint la fin du support.
Media.cuePoint	MediaDisplay, MediaPlayback	Notification indiquant que la tête de lecture a atteint un point de repère donné.
Media.playheadChange	MediaController, MediaPlayback	Diffusé par l'occurrence du composant lorsque l'utilisateur déplace le curseur de lecture ou clique sur le bouton Rembobiner ou Atteindre la fin.

Evénement	Composants	Description
Media.progress	MediaDisplay, MediaPlayback	Est généré en continu jusqu'à la fin du téléchargement du support.
Media.volume	MediaController, MediaPlayback	Diffusé lorsque l'utilisateur règle le volume.

Media.activePlayControl

S'applique à

MediaController

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.activePlayControl
```

Description

Propriété : une valeur booléenne déterminant l'état du composant MediaController lorsqu'il est chargé à l'exécution. Une valeur `true` indique que le composant MediaController devrait être en lecture à l'exécution et une valeur `false` indique qu'il est en pause à l'exécution. Cette propriété doit être définie en conjonction avec la propriété `autoPlay`, de sorte que toutes deux soient en pause ou en lecture à l'exécution. La valeur par défaut est `true`.

Exemple

L'exemple suivant indique que le contrôle sera en pause lors de son premier chargement à l'exécution.

```
monSupport.activePlayControl = false;
```

Voir aussi

Media.autoPlay

Media.addCuePoint()

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.addCuePoint(nomPointRepère, heurePointRepère)
```

Paramètres

nomPointRepère Chaîne pouvant être utilisée pour nommer le point de repère.

heurePointRepère Nombre, en secondes, indiquant le moment de diffusion d'un événement `cuePoint`.

Renvoie

Rien.

Description

Méthode : ajoute un objet point de repère à une occurrence de composant MediaPlayback ou MediaDisplay. Lorsque l'heure de la tête de lecture est égale à celle d'un point de repère, un événement `cuePoint` est diffusé.

Exemple

Le code suivant ajoute un point de repère nommé `Homerun` à `monSupport` à l'heure = `16` secondes.

```
monSupport.addCuePoint("Homerun", 16);
```

Voir aussi

`Media.cuePoint`, `Media.cuePoints`, `Media.getCuePoint()`, `Media.removeAllCuePoints()`, `Media.removeCuePoint()`

Media.aspectRatio

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.aspectRatio
```

Description

Propriété : une valeur booléenne indiquant si une occurrence de MediaDisplay ou de MediaPlayback conserve les proportions de sa vidéo pendant la lecture. Une valeur `true` indique que les proportions doivent être conservées, et une valeur `false` indique que les proportions peuvent changer pendant la lecture. La valeur par défaut est `true`.

Exemple

L'exemple suivant indique que les proportions peuvent changer lors de la lecture :

```
monSupport.aspectRatio = false;
```

Media.associateController()

S'applique à

MediaDisplay

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.associateController(nomOccurrence)
```

Paramètres

nomOccurrence Une chaîne indiquant le nom d'occurrence du composant MediaController à associer.

Renvoie

Rien.

Description

Méthode : associe une occurrence de composant MediaDisplay à une occurrence de MediaController donnée.

Si vous avez associé une occurrence de MediaController à une occurrence de MediaDisplay à l'aide de `Media.associateDisplay()`, il n'est pas nécessaire d'utiliser `Media.associateController()`.

Exemple

Le code suivant associe `monSupport` à `monContrôleur` :

```
monSupport.associateController(monContrôleur);
```

Voir aussi

`Media.associateDisplay()`

Media.associateDisplay()

S'applique à

MediaController

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.associateDisplay(nomOccurrence)
```

Paramètres

nomOccurrence Une chaîne indiquant le nom d'occurrence du composant MediaDisplay à associer.

Renvoie

Rien.

Description

Méthode : associe une occurrence de composant MediaController à une occurrence de MediaDisplay donnée.

Si vous avez associé une occurrence de MediaDisplay à une occurrence de MediaController à l'aide de `Media.associateController()`, il n'est pas nécessaire d'utiliser `Media.associateDisplay()`.

Exemple

Le code suivant associe `monSupport` à `monAffichage` :

```
monSupport.associateDisplay(monAffichage);
```

Voir aussi

`Media.associateController()`

Media.autoPlay

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.autoPlay
```

Description

Propriété : une valeur booléenne indiquant si une occurrence de MediaPlayback ou de MediaDisplay est immédiatement mise en mémoire tampon et lue. Une valeur `true` indique que le contrôle est mis en mémoire tampon et lu à l'exécution alors qu'une valeur `false` indique que le contrôle est arrêté à l'exécution. Cette propriété dépend des propriétés `contentPath` et `mediaType`. Si les propriétés `contentPath` et `mediaType` ne sont pas définies, il n'y aura pas de lecture à l'exécution. La valeur par défaut est `true`.

Exemple

L'exemple suivant indique que le contrôle ne sera pas lancé lors de son premier chargement à l'exécution.

```
monSupport.autoPlay = false;
```

Voir aussi

`Media.contentPath`, `Media.mediaType`

Media.autoSize

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.autoSize
```

Description

Propriété : une valeur booléenne indiquant comment la zone d'affichage du support du composant MediaDisplay ou MediaPlayback est redimensionnée.

Pour le composant MediaDisplay, la propriété se comporte de la manière suivante :

- Si vous définissez cette propriété sur `true`, Flash affiche le support à sa taille préférée, sans tenir compte de la taille du composant. Ceci implique que, à moins que la taille de l'occurrence MediaDisplay ne soit la même que celle du support, le support dépassera les limites de l'occurrence ou ne remplira pas la taille de l'occurrence.
- Si vous définissez cette propriété sur `false`, Flash utilise la taille de l'occurrence dans la mesure du possible, tout en respectant les proportions. Si `Media.autoSize` et `Media.aspectRatio` sont tous deux définis sur `false`, la taille exacte du composant est utilisée.

Pour le composant MediaPlayback, la propriété se comporte de la manière suivante :

- Si vous définissez cette propriété sur `true`, Flash affiche le support à sa taille préférée, sauf si la zone du support de lecture est inférieure à la taille préférée. Si tel est le cas, Flash réduit le support pour qu'il puisse s'adapter à la taille de l'occurrence et respecter les proportions. Si la taille préférée est inférieure à la zone de support de l'occurrence, une partie de la zone du support ne sera pas utilisée.
- Si vous définissez cette propriété sur `false`, Flash utilise la taille de l'occurrence dans la mesure du possible, tout en respectant les proportions. Si `Media.autoSize` et `Media.aspectRatio` sont définis sur `false`, la zone du support du composant est remplie. Cette zone est située au-dessus des contrôles (dans la disposition par défaut) et est entourée d'une marge de 8 pixels correspondant aux bords du composant.

La valeur par défaut est `true`.

Exemple

L'exemple suivant indique que le contrôle ne sera pas lu en fonction de la taille de son support :

```
monSupport.autoSize = false;
```

Voir aussi

`Media.aspectRatio`

Media.backgroundStyle

S'applique à

MediaController

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.backgroundStyle
```

Description

Propriété : une valeur `default` indique que l'arrière-plan chromé sera dessiné pour l'occurrence de MediaController, alors qu'une valeur `none` indique qu'aucun arrière-plan chromé ne sera dessiné. La valeur par défaut est `default`.

Il ne s'agit pas d'une propriété style et elle ne sera donc pas affectée par les paramètres de style.

Exemple

L'exemple suivant indique que l'arrière-plan chromé ne sera pas dessiné pour le contrôle :

```
monSupport.backgroundStyle = "none";
```

Media.bytesLoaded

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.bytesLoaded
```

Description

Propriété lecture seule : le nombre d'octets déjà chargés dans le composant pouvant être lus. La valeur par défaut est `undefined`.

Exemple

Le code suivant crée une variable appelée `PlaybackLoad` qui sera définie avec le nombre d'octets chargés dans la boucle `for`.

```
// créez une variable contenant le nombre d'octets chargés
var PlaybackLoad = monSupport.bytesLoaded;
// exécutez une fonction jusqu'à ce que la lecture soit prête
for (PlaybackLoad < 150) {
  function X();
}
```

Media.bytesTotal

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.bytesTotal
```

Description

Propriété : le nombre d'octets à charger dans le composant MediaPlayback ou MediaDisplay. La valeur par défaut est `undefined`.

Exemple

L'exemple suivant indique à l'utilisateur la taille du support à diffuser en flux continu.

```
monChampDeTexte.text = monSupport.bytesTotal;
```

Media.change

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
objetDécoute.change = function(objetEvt){
  // insérez votre code ici
}
monSupport.addEventListener("change", objetDécoute)
```

Description

Evénement : diffusé par les composants MediaDisplay et MediaPlayback pendant la lecture du support. Le pourcentage atteint peut être récupéré depuis l'occurrence de composant. Consultez l'exemple ci-dessous.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*objetEvt*) au gestionnaire. 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. L'objet événement de l'événement `Media.change` possède deux propriétés supplémentaires :

`target` Une référence à l'objet diffusé.

`type` La chaîne `"change"`, qui indique le type d'événement.

Pour plus d'informations sur les objets événement, consultez *Objets événement*, page 592.

Exemple

L'exemple suivant utilise un écouteur d'objet pour déterminer la position de la tête de lecture (`Media.playheadTime`), à partir de laquelle le pourcentage atteint peut être calculé

```
var monEcouteurDeLecteur = new Object();
monEcouteurDeLecteur.change = function(objetEvt){
  var maPosition = monLecteur.playheadTime;
  var maPositionDePourcentage = (maPosition/totalTime);
}
monLecteur.addEventListener("change", monEcouteurDeLecteur);
```

Voir aussi

`Media.playing`, `Media.pause()`

Media.click

S'applique à

MediaController, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
var monEcouteurDeSupport = new Object()
monEcouteurDeSupport.click = function(){
  // insérez votre code ici
}
monLecteur.addEventListener("click", monEcouteurDeSupport);
```

Description

Evénement : diffusé lorsque l'utilisateur clique sur le bouton de lecture/pause. Le champ détail doit être utilisé pour indiquer sur quel bouton l'utilisateur a cliqué. L'objet événement `Media.click` possède les propriétés suivantes :

`detail` La chaîne `"pause"` ou `"play"`.

`target` Une référence à l'occurrence du composant MediaController ou MediaPlayback.

`type` La chaîne `"click"`.

Exemple

L'exemple suivant ouvre une fenêtre contextuelle lorsque l'utilisateur clique sur Lire :

```
var monEcouteurDeSupport = new Object()
monEcouteurDeSupport.click = function(){
  PopUpManager.createPopup(_root, mx.containers.Window, false, {contentPath:
  movieSale});
}
monSupport.addEventListener("click", monEcouteurDeSupport);
```

Media.complete

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
ObjetDécoute.complete = function(objetEvt){
  // insérez votre code ici
}
monSupport.addEventListener("terminé", objetDécoute)
```

Description

Evénement : notification indiquant que la tête de lecture a atteint la fin du support. L'objet événement `Media.complete` possède les propriétés suivantes :

`target` Une référence à l'occurrence du composant MediaDisplay ou MediaPlayback.

`type` La chaîne `"terminé"`.

Exemple

L'exemple suivant utilise un écouteur d'objet pour indiquer le moment ou la lecture du support est terminée :

```
var monEcouteur = new Object();
monEcouteur.complete = function(objetEvt) {
    trace("le support est terminé");
};
monSupport.addEventListener("terminé", monEcouteur);
```

Media.contentPath

S'applique à

MediaController

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.contentPath
```

Description

Propriété : une chaîne contenant le chemin relatif et le nom de fichier du support devant être diffusé en flux continu et/ou lu. La méthode `Media.setMedia()` est la seule qui permet de définir cette propriété via ActionScript. La valeur par défaut est `undefined`.

Exemple

L'exemple suivant affiche le nom de l'animation lue dans un champ de texte :

```
monChampDeTexte = monSupport.contentPath;
```

Voir aussi

`Media.setMedia()`

Media.controllerPolicy

S'applique à

MediaController, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.controllerPolicy
```

Description

Propriété : détermine si le composant MediaController (ou le sous-composant du contrôleur dans le composant MediaPlayback) est masqué lorsqu'il est instancié et n'est affiché que lorsque l'utilisateur fait passer la souris sur l'état réduit du contrôleur.

Les valeurs possibles pour cette propriété sont les suivantes :

- `"on"` indique que les contrôles sont toujours développés.
- `"off"` indique que les contrôles sont toujours réduits.
- `"auto"` indique que le contrôle reste à l'état réduit jusqu'à ce que l'utilisateur déplace la souris sur la zone active. La zone active correspond à la zone dans laquelle est dessiné le contrôle réduit. Le contrôle reste développé jusqu'à ce que la souris quitte cette zone.

Remarque : La zone active se développe et se réduit avec le contrôleur.

Exemple

L'exemple suivant maintient le contrôleur ouvert en permanence :

```
monSupport.controllerPolicy = "on";
```

Media.controlPlacement

S'applique à

MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.controlPlacement
```

Description

Propriété : détermine l'endroit où la portion du contrôleur du composant MediaPlayback est positionnée par rapport à son affichage. Les valeurs possibles sont `"top"` (haut), `"bottom"` (bas), `"left"` (gauche) et `"right"` (droite). La valeur par défaut est `"bottom"`.

Exemple

Dans l'exemple suivant, la portion du contrôleur du composant MediaPlayback sera située sur le côté droit :

```
monSupport.controlPlacement = "right";
```

Media.cuePoint

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
objetDécoute.cuePoint = function(objetEvt){
  // insérez votre code ici
}
monSupport.addEventListener("pointRepère", objetDécoute)
```

Description

Evénement : notification indiquant que la tête de lecture a atteint le point de repère. L'objet événement `Media.cuePoint` possède les propriétés suivantes :

`name` Une chaîne indiquant le nom du point de repère.

`time` Un nombre, exprimé en images ou en secondes, indiquant le moment où le point de repère est atteint.

`target` Une référence à l'objet point de repère.

`type` La chaîne `"pointRepère"`.

Exemple

L'exemple suivant utilise un écouteur d'objet pour déterminer le moment où un point de repère a été atteint :

```
var monEcouteurPointRepère = new Object();
monEcouteurPointRepère.cuePoint = function(objetEvt){
  trace("écouté " + eventObject.type + ", " + eventObject.target);
}
maLecture.addEventListener("pointRepère", monEcouteurPointRepère);
```

Voir aussi

`Media.addCuePoint()`, `Media.cuePoints`, `Media.getCuePoint()`

Media.cuePoints

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.cuePoints[N]
```

Description

Propriété : un tableau d'objets point de repère ayant été affectés à une occurrence de composant MediaPlayback ou MediaDisplay. Dans le tableau, chaque objet point de repère peut avoir un nom, un temps en secondes ou en images et une propriété de lecteur (qui est le nom d'occurrence du composant auquel il est associé). La valeur par défaut est un tableau vide `[]`.

Exemple

L'exemple suivant supprime le troisième point de repère en cas de lecture d'aperçu d'une action :

```
if(maVariable == actionPreview) {
  monSupport.removeCuePoint(monSupport.cuePoints[2]);
}
```

Voir aussi

`Media.addCuePoint()`, `Media.getCuePoint()`, `Media.removeCuePoint()`

Media.displayFull()

S'applique à

MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.displayFull()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : définit l'occurrence du composant MediaPlayback en mode plein écran. En d'autres termes, le composant se développe pour remplir la totalité de la scène. Pour que le composant retrouve sa taille normale, utilisez `Media.displayNormal()`.

Exemple

Le code suivant force le composant à se développer pour s'adapter à la taille de la scène :

```
monSupport.displayFull();
```

Voir aussi

`Media.displayNormal()`

Media.displayNormal()

S'applique à

MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.displayNormal()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : redéfinit l'occurrence de MediaPlayback sur sa taille normale après l'utilisation d'une méthode `Media.displayFull()`.

Exemple

Le code suivant renvoie un composant MediaPlayback à sa taille originale :

```
monSupport.displayNormal();
```

Voir aussi

`Media.displayFull()`

Media.getCuePoint()

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.getCuePoint(nomPointRepère)
```

Paramètres

Aucun.

Renvoie

nomPointRepère La chaîne fournie lors de l'utilisation de `Media.addCuePoint()`.

Description

Méthode : renvoie un objet point de repère en fonction de son nom de point de repère.

Exemple

Le code suivant récupère un point de repère appelé `monPointDeRepère`.

```
monSupport.removeCuePoint(monSupport.getCuePoint("monPointDeRepère"));
```

Voir aussi

`Media.addCuePoint()`, `Media.cuePoint`, `Media.cuePoints`, `Media.removeCuePoint()`

Media.horizontal

S'applique à

MediaController

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.horizontal
```

Description

Propriété : détermine si le composant MediaController est affiché verticalement ou horizontalement. Une valeur `true` indique que le composant est affiché horizontalement et une valeur `false` indique une orientation verticale. Lorsque la valeur est `false`, la tête de lecture et l'indicateur de progression du chargement se déplacent de bas en haut. La valeur par défaut est `true`.

Exemple

L'exemple suivant affiche le composant MediaController dans une position verticale :

```
monSupport.horizontal = false;
```

Media.mediaType

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.mediaType
```

Description

Propriété : contient la valeur du type de support à lire. Les deux choix possibles sont les formats FLV et MP3. La valeur par défaut est `"FLV"`. Consultez « Importation des fichiers FLV (Macromedia Flash Video) », dans le guide Utilisation de Flash de l'aide.

Exemple

L'exemple suivant détermine le type de support actuel en cours de lecture :

```
var supportActuel = monSupport.mediaType;
```

Voir aussi

`Media.setMedia()`

Media.pause()

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.pause()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : met la tête de lecture en pause à l'emplacement actuel.

Exemple

Le code suivant met la lecture en pause.

```
monSupport.pause()
```

Media.play()

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.play(pointDépart)
```

Paramètres

pointDépart Une valeur entière non négative indiquant le point de départ (en secondes) de la lecture du support.

Renvoie

Rien.

Description

Méthode : lit le support associé à l'occurrence de composant à un point de départ donné. La valeur par défaut est la valeur actuelle de `playheadTime`.

Exemple

Le code suivant indique que la lecture du composant de support doit commencer à 120 secondes :

```
monSupport.play(120);
```

Voir aussi

`Media.pause()`

Media.playheadChange

S'applique à

MediaController, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
objetDécoute.playheadChange = function(objetEvt){
  // insérez votre code ici
}
monSupport.addEventListener("playheadChange", objetDécoute)
```

Description

Evénement : diffusé par le composant MediaController ou MediaPlayback lorsque l'utilisateur déplace le curseur de lecture ou clique sur le bouton Rembobiner ou Atteindre la fin. L'objet événement `Media.playheadChange` possède les propriétés suivantes :

`detail` Un nombre indiquant le pourcentage de support qui a été lu.

`type` La chaîne `"playheadChange"`.

Exemple

L'exemple suivant envoie le pourcentage lu au panneau de sortie lorsque l'utilisateur arrête de déplacer la tête de lecture :

```
var écouteurContrôle = new Object();
écouteurContrôle.playheadChange = function(objetEvt){
  trace(eventObject.detail);
}
monSupport.addEventListener("playheadChange", écouteurContrôle);
```

Media.playheadTime

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.playheadTime
```

Description

Propriété : contient la position actuelle de la tête de lecture (en secondes) pour le scénario du support en cours de lecture. La valeur par défaut est définie sur l'emplacement de la tête de lecture.

Exemple

L'exemple suivant définit une variable à l'emplacement de la tête de lecture, indiqué en secondes :

```
var maTêteDeLecture = monSupport.playheadTime;
```

Media.playing

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.playing
```

Description

Propriété lecture seule : renvoie une valeur booléenne indiquant si le support est en cours de lecture. Une valeur `true` indique que le support est en cours de lecture. Une valeur `false` indique que le support est mis en pause par l'utilisateur.

Exemple

Le code suivant détermine si le support est en cours de lecture ou en pause :

```
if(monSupport.playing == true){
  une fonction;
}
```

Voir aussi

`Media.change`

Media.preferredHeight

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.preferredHeight
```

Description

Propriété : définie en fonction de la valeur de la hauteur par défaut d'un fichier FLV. Cette propriété s'applique uniquement aux supports FLV car la hauteur des fichiers MP3 est fixe. Cette propriété peut être utilisée pour définir les paramètres de hauteur et de largeur (ainsi qu'une marge pour le composant lui-même). La valeur par défaut est `undefined` si aucun support FLV n'est défini.

Exemple

L'exemple suivant redimensionne une occurrence de MediaPlayback par rapport à l'occurrence qu'elle lit et tient compte de la marge de pixels nécessaire à l'occurrence de composant :

```
if(maLecture.contentPath = !undefined){
  var hauteurSupport = maLecture.preferredHeight;
  var largeurSupport = maLecture.preferredWidth;
  maLecture.setSize((largeurSupport + 20), (hauteurSupport + 70));
  }
```

Media.preferredWidth

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.preferredWidth
```

Description

Propriété : définie en fonction de la valeur de la largeur par défaut d'un fichier FLV. La valeur par défaut est `undefined`.

Exemple

L'exemple suivant définit la largeur désirée de la variable `largeurSupport` :

```
var largeurSupport = monSupport.preferredWidth;
```

Media.progress

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
objetDécoute.progress = function(objetEvt){
  // insérez votre code ici
}
monSupport.addEventListener("progress", objetDécoute)
```

Description

Evénement : est généré en continu jusqu'à la fin du téléchargement du support. L'objet événement `Media.progress` possède les propriétés suivantes :

`target` Une référence à l'occurrence du composant MediaDisplay ou MediaPlayback.

`type` La chaîne `"progress"`.

Exemple

L'exemple suivant écoute la progression :

```
var monEcouteurDeProgression = new Object();
monEcouteurDeProgression.progress = function(){
  // Faites clignoter lightMovieClip pendant la progression
  var lumièreVisible = lightMovieClip.visible;
  lightMovieClip.visible = !lightVisible;
}
```

Media.removeAllCuePoints()

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.removeAllCuePoints()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : supprime tous les objets point de repère associés à une occurrence de composant.

Exemple

Le code suivant supprime tous les objets point de repère :

```
monSupport.removeAllCuePoints();
```

Voir aussi

`Media.addCuePoint()`, `Media.cuePoints`, `Media.removeCuePoint()`

Media.removeCuePoint()

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.removeCuePoint(pointRepère)
```

Paramètres

`pointRepère` Une référence à un objet point de repère ayant été précédemment affecté par le biais de `Media.addCuePoint()`.

Renvoie

Rien.

Description

Méthode : supprime un objet point de repère spécifique associé à une occurrence de composant.

Exemple

Le code suivant supprime un point de repère appelé `monPointRepère` :

```
monSupport.removeCuePoint(getCuePoint("monPointRepère"));
```

Voir aussi

`Media.addCuePoint()`, `Media.cuePoints`, `Media.removeAllCuePoints()`

Media.setMedia()

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.setMedia(contentPath, mediaType)
```

Paramètres

contentPath Une chaîne indiquant le chemin et le nom de fichier du support à lire.

mediaType Une chaîne utilisée pour définir le type de support sur FLV ou MP3. Ce paramètre est facultatif.

Renvoie

Rien.

Description

Méthode : définit le type de support et le chemin vers le type de support spécifié à l'aide d'un argument d'URL. La valeur par défaut pour *contentPath* est undefined.

Cette méthode est la seule qui permette de définir le chemin du contenu et le type du support pour les composants MediaPlayback et MediaDisplay.

Exemple

Le code suivant fournit un nouveau support à lire pour une occurrence de composant :

```
monSupport.setMedia("http://www.RogerMoore.com/moonraker.flv", "FLV");
```

Media.stop()

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.stop()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : arrête la tête de lecture et la déplace à la position 0, qui correspond au début du support.

Exemple

Le code suivant arrête la tête de lecture et la déplace à l'heure = 0.

```
monSupport.stop()
```

Media.totalTime

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.totalTime
```

Description

Propriété : la durée totale du support, en secondes. Etant donné que le format de fichier FLV ne fournit pas sa durée de lecture à un composant de support tant que son chargement n'est pas terminé, il est nécessaire d'ajouter manuellement `Media.totalTime` de sorte que le curseur de lecture puisse refléter précisément la durée de lecture réelle du support. La valeur par défaut pour les fichiers MP3 est la durée de lecture du support. Pour les fichiers FLV, la valeur par défaut est `undefined`.

Il est impossible de définir cette propriété pour les fichiers MP3 car cette information est contenue dans l'objet Sound.

Exemple

L'exemple suivant définit la durée de lecture en secondes pour le support FLV :

```
monSupport.totalTime = 151;
```

Media.volume

S'applique à

MediaDisplay, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
monSupport.volume
```

Description

Propriété : stocke la valeur entière du paramètre volume, qui peut aller de 0 à 100. La valeur par défaut pour cette propriété est 75.

Exemple

L'exemple suivant définit le volume maximum pour la lecture du support :

```
monSupport.volume = 100;
```

Voir aussi

`Media.volume`, `Media.pause()`

Media.volume

S'applique à

MediaController, MediaPlayback

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
objetDécoute.volume = function(objetEvt){
  // insérez votre code ici
}
monSupport.addEventListener("volume", objetDécoute)
```

Description

Evénement : diffusé lorsque la valeur du volume est réglée par l'utilisateur. L'objet événement `Media.volume` possède les propriétés suivantes :

`detail` Une valeur entière comprise entre 0 et 100 qui représente le niveau de volume.

`type` La chaîne `"volume"`.

Exemple

L'exemple suivant informe l'utilisateur que le volume est en cours de réglage.

```
var monEcouteurDeVol = new Object();
monEcouteurDeVol.volume = function(){
   monChampDeTexte.text = "Volume réglé !";
}
monSupport.addEventListener("volume", monEcouteurDeVol)
```

Voir aussi

`Media.volume`

Composant Menu (Flash Professionnel uniquement)

Le composant Menu permet à un utilisateur de sélectionner un élément depuis un menu déroulant, tout comme le menu Fichier ou Edition de la plupart des applications logicielles.

Un menu s'ouvre généralement dans une application lorsqu'un utilisateur survole un activateur de menu de type bouton ou clique sur ce dernier. Vous pouvez également écrire un script indiquant à un composant Menu de s'ouvrir lorsqu'un utilisateur appuie sur une touche spécifique.

Les composants Menu sont toujours créés dynamiquement à l'exécution. Vous devez ajouter le composant au document à partir du panneau Composants, puis le supprimer pour l'ajouter dans la bibliothèque. Utilisez le code suivant pour créer un menu avec ActionScript :

```
var monMenu = mx.controls.Menu.createMenu(parent, menuDataProvider);
```

Utilisez le code suivant pour ouvrir un menu dans une application :

```
monMenu.show(x, y);
```

Un événement `menuShow` est diffusé à l'ensemble des écouteurs des occurrences de Menu avant que le menu ne soit rendu, afin de pouvoir mettre à jour l'état des éléments de menu. De la même façon, un événement `menuHide` est diffusé juste après qu'une occurrence de menu est masquée.

Les éléments d'un menu sont décrits avec XML. Pour plus d'informations, consultez *Présentation du composant Menu : affichage et données*, page 388.

Vous ne pouvez pas rendre le composant Menu accessible aux lecteurs d'écran.

Interaction avec le composant Menu (Flash Professionnel uniquement)

Vous pouvez utiliser la souris et le clavier pour interagir avec un composant Menu.

Une fois qu'un menu est ouvert, il reste visible jusqu'à ce qu'il soit fermé par un script ou jusqu'à ce que l'utilisateur clique avec la souris en dehors du menu ou à l'intérieur d'un élément activé.

Vous pouvez sélectionner un élément de menu en cliquant sur ce dernier, excepté pour les types d'éléments de menu suivants :

- Les éléments désactivés ou les séparateurs Les survols et clics n'ont aucun effet (le menu reste visible).
- Les ancres d'un sous-menu Les survols activent le sous-menu, les clics n'ont aucun effet. Le survol de tout élément autre que ceux appartenant au sous-menu ferme le sous-menu.

Lorsqu'un élément est sélectionné, un événement `Menu.change` est envoyé à l'ensemble des écouteurs du menu, le menu est masqué et les actions suivantes se produisent, en fonction du type d'élément :

- `check` L'attribut `selected` de l'élément est activé.
- `radio` L'élément devient la sélection actuelle de son groupe de boutons radio.

Le déplacement de la souris déclenche des événements `Menu.rollOut` et `Menu.rollOver`.

Le fait d'appuyer sur le bouton de la souris en dehors du menu ferme le menu et déclenche un événement `Menu.menuHide`.

Le fait de relâcher le bouton de la souris dans un élément activé affecte les types d'éléments comme suit :

- `check` L'attribut `selected` de l'élément est activé.
- `radio` L'attribut `selected` de l'élément est défini sur `true` et l'attribut `selected` de l'élément précédemment sélectionné dans le groupe de boutons radio est défini sur `false`. La propriété `selection` de l'objet groupe de boutons radio correspondant est définie pour faire référence à l'élément de menu sélectionné.
- undefined et le parent d'un menu hiérarchique La visibilité du menu hiérarchique est activée.

Lorsqu'une occurrence de Menu a le focus via un clic ou l'utilisation de la tabulation, vous pouvez utiliser les touches suivantes pour la contrôler :

Touche	Description
Flèche vers le bas Flèche vers le haut	Déplace la sélection de bas en haut dans les lignes du menu. La sélection boucle dans la première ou la dernière ligne.
Flèche vers la droite	Ouvre un sous-menu ou déplace la sélection vers le menu suivant dans une barre de menus (si une barre de menus est disponible).
Flèche vers la gauche	Ferme un sous-menu et renvoie le focus au menu parent (si un menu parent est disponible) ou déplace la sélection vers le menu précédent dans une barre de menus (si une barre de menus est disponible).
Entrée	Ouvre un sous-menu, ou clique et relâche le bouton de la souris sur une ligne si aucun sous-menu n'existe.

Remarque : Si un menu est ouvert, vous pouvez appuyer sur la touche de tabulation pour quitter le menu. Vous devez effectuer une sélection ou quitter le menu en appuyant sur Echap.

Utilisation du composant Menu (Flash Professionnel uniquement)

Vous pouvez utiliser le composant Menu pour créer des menus d'options sélectionnables individuellement semblables aux menus Fichier et Edition de la plupart des applications logicielles. Vous pouvez également utiliser le composant Menu pour créer des menus contextuels qui s'affichent lorsque l'utilisateur appuie sur une zone sensible ou une touche de modification. Utilisez le composant Menu avec le composant MenuBar pour créer une barre de menus horizontale contenant des menus qui se développent sous chaque élément de la barre de menus.

Tout comme les menus d'ordinateur de bureau standard, le composant Menu supporte les éléments de menu dont les fonctions correspondent aux catégories générales suivantes :

Activateurs de commandes Ces éléments déclenchent des événements. Vous devez écrire du code pour gérer ces événements.

Ancres de sous-menus Ces éléments sont des ancres qui ouvrent des sous-menus.

Boutons radio Ces éléments fonctionnent par groupe. Vous ne pouvez sélectionner qu'un élément à la fois.

Eléments de case à cocher Ces éléments représentent une valeur booléenne (`true` ou `false`).

Séparateurs Ces éléments fournissent une ligne horizontale unique qui divise les éléments d'un menu en différents groupes visuels.

Présentation du composant Menu : affichage et données

Au niveau de sa conception, le composant Menu est composé d'un modèle de données et d'une vue qui affiche les données. La classe Menu est la vue et elle contient les méthodes de configuration visuelle. La *Classe MenuDataProvider* ajoute des méthodes à l'objet prototype XML global (tout comme la classe DataProvider le fait pour l'objet Array). Ces méthodes vous permettent de créer des fournisseurs de données de façon externe et de les ajouter à plusieurs occurrences de menu. Le fournisseur de données diffuse toutes les modifications à l'ensemble de ses vues client. Pour plus d'informations, consultez *Classe MenuDataProvider*, page 410.

Une occurrence de Menu est un ensemble hiérarchique d'éléments XML correspondant aux éléments de menu individuels. Les attributs définissent le comportement et l'apparence de l'élément de menu correspondant à l'écran. Cet ensemble peut facilement être traduit depuis et vers le langage XML, qui est utilisé pour décrire les menus (la balise de menu) et les éléments (la balise menuitem). La classe XML ActionScript intégrée est la base du modèle sous-jacent du composant Menu.

Un menu simple contenant deux éléments peut être décrit dans XML avec deux sous-éléments d'élément de menu :

```
<Menu>
  <menuitem label="Haut" />
  <menuitem label="Bas" />
</menu>
```

Remarque : Les noms de balise des nœuds XML (`menu` et `menuitem`) ne sont pas importants. Les attributs et leurs relations d'imbrication sont utilisés dans le menu.

A propos des menus hiérarchiques

Pour créer des menus hiérarchiques, imbriquez des éléments XML dans un élément XML parent de la manière suivante :

```
<Menu>
    <menuitem label="ElémentMenu A" >
      <menuitem label="ElémentSousMenu 1-A" />
      <menuitem label="ElémentSousMenu 2-A" />
    </menuitem>
    <menuitem label="ElémentMenu B" >
      <menuitem label="ElémentSousMenu 1-B" />
      <menuitem label="ElémentSousMenu 2-B" />
    </menuitem>
</menu>
```

Remarque : Ceci convertit l'élément du menu parent en ancre de menu déroulant, pour qu'il ne génère pas d'événement une fois sélectionné.

A propos des attributs XML d'élément de menu

Les attributs de l'élément XML d'un élément de menu déterminent ce qui est affiché, le comportement de l'élément de menu et la façon dont il est exposé à ActionScript. Le tableau suivant décrit les attributs d'un élément de menu XML :

Nom de l'attribut	Type	Valeur par défaut	Description
`label`	String	`undefined`	Le texte affiché pour représenter un élément de menu. Cet attribut est requis pour tous les types d'élément, excepté pour les séparateurs.
`type`	`separator`, `check`, `radio`, `normal` ou undefined	`undefined`	Le type d'élément de menu : `separator`, `check box`, radio `button` ou `normal` (une commande ou un activateur de sous-menu). Si cet attribut n'existe pas, la valeur par défaut est `normal`.
`icon`	String	`undefined`	L'identificateur de liaison d'un actif d'image. Cet attribut n'est pas requis. Cet attribut n'est pas disponible pour les types `check`, `radio`, ou `separator`.
`instanceName`	String	`undefined`	Un identificateur que vous pouvez utiliser pour faire référence à l'occurrence d'élément de menu depuis l'occurrence de menu racine. Par exemple, un élément de menu nommé *jaune* peut être référencé en tant que `monMenu.jaune`. Cet attribut n'est pas requis.
`groupName`	String	`undefined`	Un identifiant que vous pouvez utiliser pour associer plusieurs boutons radio au sein d'un groupe radio et pour exposer l'état d'un groupe radio à partir de l'occurrence de menu racine. Par exemple, un groupe radio nommé *couleurs* peut être référencé en tant que `monMenu.couleurs`. Cet attribut est uniquement requis pour le type `radio`.
`selected`	`false`, `true` ou `false` ou `true` (une chaîne ou une valeur booléenne)	`false`	Une valeur booléenne indiquant si un élément `check` ou `radio` est activé (`true`) ou désactivé (`false`). Cet attribut n'est pas requis.
`enabled`	`false`, `true` ou `false` ou `true` (une chaîne ou une valeur booléenne)	`true`	Une valeur booléenne indiquant si l'élément de menu peut être sélectionné (`true`) ou non (`false`). Cet attribut n'est pas requis.

A propos des types d'élément de menu

Il existe quatre types d'élément de menu, spécifiés par l'attribut `type` :

```
<Menu>
    <menuitem label="Elément normal" />
    <menuitem type="separator" />
    <menuitem label="Elément Case à cocher" type="check"
  instanceName="cocher_1"/>
    <menuitem label="Elément Bouton radio" type="radio"
  groupName="groupeRadio_1" />
</menu>
```

Eléments de menu normaux

L'`élément de menu normal` ne possède pas d'attribut `type`, ce qui signifie que son attribut `type` prend par défaut la valeur `normal`. Les éléments normaux peuvent être des activateurs de commande ou des activateurs de sous-menu, selon qu'ils comportent des sous-éléments imbriqués ou non.

Eléments de menu séparateurs

Les éléments de menu dont l'attribut `type` est défini sur `separator` servent de séparateurs visuels dans le menu. Le code XML suivant crée les trois éléments de menu Haut, Milieu et Bas, ainsi que des séparateurs entre chacun d'entre eux :

```
<Menu>
    <menuitem label="Haut" />
    <menuitem type="separator" />
    <menuitem label="Milieu" />
    <menuitem type="separator" />
    <menuitem label="Bas" />
</menu>
```

Tous les éléments séparateurs sont désactivés. Le fait de cliquer ou de passer la souris sur un séparateur n'a aucun effet.

Eléments de menu Case à cocher

Les éléments de menu dont l'attribut `type` est défini sur `check` jouent le rôle de case à cocher dans le menu ; lorsque l'attribut `selected` est défini sur `true`, une case à cocher apparaît en regard de l'étiquette de l'élément de menu. Lorsqu'un élément case à cocher est sélectionné, son état change immédiatement et un événement `change` est diffusé à tous les écouteurs du menu racine. L'exemple suivant définit trois éléments de menu de type case à cocher :

```
<Menu>
    <menuitem label="Pommes" type="check" instanceName="acheterDesPommes"
  selected="true" />
    <menuitem label="Oranges" type="check" instanceName="acheterDesOranges"
  selected="false" />
    <menuitem label="Bananes" type="check" instanceName="acheterDesBananes"
  selected="false" />
</menu>
```

Dans ActionScript, les noms d'occurrence vous permettent d'accéder directement aux éléments de menu à partir du menu, comme dans l'exemple suivant :

```
monMenu.setMenuItemSelected(monMenu.acheterdespommes, true);
monMenu.setMenuItemSelected(monMenu.acheterdesoranges, false);
```

Remarque : Pour modifier l'attribut `selected`, seule la méthode `setMenuItemSelected(item, b)` doit être utilisée. Vous pouvez examiner directement l'attribut `selected`, mais celui-ci renvoie une valeur de chaîne `true` ou `false`.

Eléments de menu Bouton radio

Les éléments de menu dont l'attribut `type` est défini sur `radio` peuvent être groupés de sorte que seul un élément puisse être sélectionné à la fois. Pour créer un groupe radio, affectez la même valeur à l'attribut `groupName` de tous les éléments de menu à inclure dans le groupe, comme dans l'exemple suivant :

```
<Menu>
    <menuitem label="Centre" type="radio" groupName="groupe_alignement"
       instanceName="élément_centre"/>
    <menuitem type="separator" />
    <menuitem label="Haut"    type="radio" groupName="groupe_alignement" />
    <menuitem label="Bas" type="radio" groupName="groupe_alignement" />
    <menuitem label="Droite"  type="radio" groupName="groupe_alignement" />
    <menuitem label="Gauche"   type="radio" groupName="groupe_alignement" />
</menu>
```

Lorsque l'utilisateur sélectionne l'un des éléments, la sélection en cours change automatiquement et un événement `change` est diffusé à tous les écouteurs du menu racine. Dans ActionScript, la propriété `sélection` permet de faire référence à l'élément actuellement sélectionné dans le groupe radio, comme dans l'exemple suivant :

```
var élémentMenuSélectionné = monMenu.groupe_alignement.sélection;
monMenu.groupe_alignement = monMenu.élément_centre;
```

Chaque valeur `groupName` doit être unique dans le domaine de l'occurrence de menu racine.

Remarque : Pour modifier l'attribut `selected`, seule la méthode `setMenuItemSelected(item, b)` doit être utilisée. Vous pouvez examiner directement l'attribut `selected`, mais celui-ci renvoie une valeur de chaîne `true` ou `false`.

Exposition des éléments de menu à ActionScript

Vous pouvez utiliser l'attribut `instanceName` pour affecter un identifiant unique à chaque élément de menu et ainsi pouvoir y accéder directement à partir du menu racine. Par exemple, le code XML suivant définit les attributs `instanceName` de chaque élément de menu :

```
<Menu>
    <menuitem label="Elément 1" instanceName="élément_1" />
    <menuitem label="Elément 2" instanceName="élément_2" >
      <menuitem label="Sous-élément A" instanceName="sous_élément_A" />
      <menuitem label="Sous-élément B" instanceName="sous_élément_B" />
    </menuitem>
</menu>
```

Vous pouvez utiliser ActionScript pour accéder aux occurrences d'objet correspondantes et à leurs attributs directement depuis le composant menu, comme dans l'exemple suivant :

```
var unElémentDeMenu = monMenu.élément_1;
monMenu.setMenuItemEnabled(élément_2, true);
var uneEtiquette = monMenu.sous_élément_A.label;
```

Remarque : Chaque attribut `instanceName` doit être unique dans le domaine de l'occurrence de menu racine (y compris dans tous les sous-menus du menu racine).

A propos des propriétés d'objet d'initialisation

Le paramètre *objetInit* (objet d'initialisation) joue un rôle fondamental dans la création de la disposition du composant Menu. Le paramètre *objetInit* est un objet possédant des propriétés. Chaque propriété représente l'un des attributs XML possibles d'un élément de menu (pour obtenir la description des propriétés autorisées dans le paramètre *objetInit*, consultez *A propos des attributs XML d'élément de menu*, page 389).

Le paramètre *objetInit* est utilisé dans les méthodes suivantes :

- `Menu.addMenuItem()`
- `Menu.addMenuItemAt()`
- `MenuDataProvider.addMenuItem()`
- `inboxMenuDataProvider.addMenuItemAt()`

L'exemple suivant crée un paramètre *objetInit* avec deux propriétés, `label` et `instanceName` :

```
var i = monMenu.addMenuItem({label:"monElémentDeMenu",
  instanceName:"monPremierElément"});
```

Vous pouvez conjuguer plusieurs propriétés pour créer un type particulier d'élément de menu. Vous affectez des propriétés spécifiques pour créer certains types d'éléments de menu (normal, séparateur, case à cocher ou bouton radio).

Pour initialiser un élément de menu normal, par exemple, utilisez le paramètre *objetInit* suivant :

```
monMenu.addMenuItem({label:"monElémentDeMenu", enabled:true, icon:"monIcône",
  instanceName:"monPremierElément"});
```

Pour initialiser un élément de menu séparateur, utilisez le paramètre *objetInit* suivant :

```
monMenu.addMenuItem({type:"separator"});
```

Pour initialiser un élément de menu case à cocher, utilisez le paramètre *objetInit* suivant :

```
monMenu.addMenuItem({type:"check", label:"maCaseACocher", enabled:false,
  selected:true, instanceName:"maPremièreCaseACocher"})
```

Pour initialiser un élément de menu bouton radio, utilisez le paramètre *objetInit* suivant :

```
monMenu.addMenuItem({type:"radio", label:"monBoutonRadio1", enabled:true,
  selected:false, groupName:"monGroupeRadio"
  instanceName:"monPremierBoutonRadio"})
```

Il est important de noter que les attributs `instanceName`, `groupName` et `type` d'un élément de menu doivent être considérés comme des éléments en lecture seule. Définissez-les uniquement au moment où vous créez un élément (par exemple, dans un appel de `addMenuItem()`). Si vous modifiez ces attributs par la suite, ils risquent de produire des résultats imprévisibles.

Paramètres du composant Menu

Il n'existe pas de paramètres de création pour le composant Menu.

Vous pouvez rédiger du code ActionScript pour contrôler le composant Menu à l'aide de ses propriétés, méthodes et événements. Pour plus d'informations, consultez *Classe Menu (Flash Professionnel uniquement)*, page 396.

Création d'une application avec le composant Menu

Dans l'exemple ci-dessous, un développeur crée une application et utilise le composant Menu pour exposer quelques-unes des commandes que les utilisateurs peuvent émettre, telles que Ouvrir, Fermer, Enregistrer, etc.

Pour créer une application avec le composant Menu :

1 Choisissez Fichier > Nouveau pour créer un document Flash.

2 Faites glisser le composant Menu du panneau Composants jusqu'à la scène et supprimez-le.

Le composant Menu est ajouté à la bibliothèque, mais pas à l'application. Les menus sont créés dynamiquement à l'aide d'ActionScript.

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

Cliquez sur le bouton pour activer le menu.

4 Dans l'inspecteur des propriétés, donnez au bouton le nom d'occurrence **BtnCommande** et la propriété text **Commandes**.

5 Dans le panneau Actions de la première image, entrez le code suivant pour ajouter un écouteur d'événement `click` à l'occurrence `BtnCommande` :

```
var écouteur = new Object();
écouteur.click = function(objEvt) {
    var bouton = objEvt.target;
  if(bouton.menu == undefined) {
        // Créer une occurrence de menu et ajouter quelques éléments
        bouton.menu = mx.controls.Menu.createMenu();
        bouton.menu.addMenuItem("Ouvrir");
        bouton.menu.addMenuItem("Fermer");
        bouton.menu.addMenuItem("Enregistrer");
        bouton.menu.addMenuItem("Rétablir");
        // Ajouter un écouteur de l'événement change pour détecter les
  sélections
        var écouteurChange = new Object();
        écouteurChange.change = function(événement) {
      var élément = événement.menuItem;
      trace("Elément sélectionné :  " + élément.attributes.label);
        }
        bouton.menu.addEventListener("change", écouteurChange);
    }
  bouton.menu.show(bouton.x, bouton.y + bouton.height);
}
btnCommande.addEventListener("click", écouteur);
```

6 Choisissez Contrôle > Tester l'animation.

Cliquez sur le bouton Commandes pour afficher le menu. Sélectionnez des éléments de menu pour visualiser les éléments sélectionnés tels qu'ils ont été consignés par les actions `trace` dans le panneau de sortie.

Pour utiliser les données XML d'un serveur afin de créer et d'alimenter un menu :

1 Choisissez Fichier > Nouveau pour créer un document Flash.

2 Faites glisser le composant Menu du panneau Composants jusqu'à la scène et supprimez-le.

Le composant Menu est ajouté à la bibliothèque, mais pas à l'application. Les menus sont créés dynamiquement à l'aide d'ActionScript.

3 Dans le panneau Actions, ajoutez le code suivant à la première image pour créer un menu et lui ajouter quelques éléments :

```
var monMenu = mx.controls.Menu.createMenu();
// Importer un fichier XML
var loader = new XML();
loader.menu = monMenu;
loader.ignoreWhite = true;
loader.onLoad = function(succès) {
  // Transmettre les données au menu lorsqu'elles arrivent
  if (succès) {
    this.menu.dataProvider = this.firstChild;
  }
};
loader.load(url);
```

Remarque : Les éléments de menu sont décrits par les enfants du premier enfant du document XML.

4 Choisissez Contrôle > Tester l'animation.

Pour utiliser une chaîne XML bien construite afin de créer et d'alimenter un menu :

1 Choisissez Fichier > Nouveau pour créer un document Flash.

2 Faites glisser le composant Menu du panneau Composants jusqu'à la scène et supprimez-le.

Le composant Menu est ajouté à la bibliothèque, mais pas à l'application. Les menus sont créés dynamiquement à l'aide d'ActionScript.

3 Dans le panneau Actions, ajoutez le code suivant à la première image pour créer un menu et lui ajouter quelques éléments :

```
// Créer une chaîne XML contenant une définition de menu
var s = "";
s += "<menu>";
s += "<menuitem label='Annuler' />";
s += "<menuitem type='separator' />";
s += "<menuitem label='Couper' />";
s += "<menuitem label='Copier' />";
s += "<menuitem label='Coller' />";
s += "<menuitem label='Effacer' />";
s += "<menuitem type='separator' />";
s += "<menuitem label='Sélectionner tout' />";
s += "</menu>";
// Créer un objet XML à partir de la chaîne
var xml = new XML(s);
xml.ignoreWhite = true;
// Créer un menu à partir du premier enfant de l'objet XML
var monMenu = mx.controls.Menu.createMenu(_root, xml.firstChild);
```

4 Choisissez Contrôle > Tester l'animation.

Pour utiliser la classe MenuDataProvider pour créer et alimenter un menu :

1 Choisissez Fichier > Nouveau pour créer un document Flash.

2 Faites glisser le composant Menu du panneau Composants jusqu'à la scène et supprimez-le.

Le composant Menu est ajouté à la bibliothèque, mais pas à l'application. Les menus sont créés dynamiquement à l'aide d'ActionScript.

3 Dans le panneau Actions, ajoutez le code suivant à la première image pour créer un menu et lui ajouter quelques éléments :

```
// Créer un objet XML qui servira de jeu préconfiguré
var xml = new XML();

// Le prochain élément créé n'apparaîtra pas dans le menu.
// L'appel de la méthode 'createMenu' (ci-dessous) attend la
// réception d'un élément racine dont les enfants deviendront
// les éléments du menu.  Il s'agit simplement d'une méthode pour créer
facilement cet
// élément racine et lui donner un nom
// pratique.
var élémentDeMenu = xml.addMenuItem("Edition");

// Ajouter les éléments de menu
élémentDeMenu.addMenuItem({label:"Annuler"});
élémentDeMenu.addMenuItem({type:"separator"});
élémentDeMenu.addMenuItem({label:"Couper"});
élémentDeMenu.addMenuItem({label:"Copier"});
élémentDeMenu.addMenuItem({label:"Coller"});
élémentDeMenu.addMenuItem({label:"Effacer", enabled:"false"});
élémentDeMenu.addMenuItem({type:"separator"});
élémentDeMenu.addMenuItem({label:"Sélectionner tout"});
// Créer l'objet Menu
var laCommandeMenu = mx.controls.Menu.createMenu(_root, élémentDeMenu);
```

4 Choisissez Contrôle > Tester l'animation.

Personnalisation du composant Menu

La taille horizontale du menu s'ajuste automatiquement pour contenir le texte le plus long. Vous pouvez également appeler la méthode `setSize()` pour définir la taille du composant. La taille des icônes ne doit pas dépasser 16 pixels par 16 pixels.

Utilisation de styles avec le composant Menu

Vous pouvez appeler la méthode `setStyle()` pour modifier le style du menu, de ses éléments et de ses sous-menus. Un composant Menu supporte les styles de halo suivants :

Style	Description
`themeColor`	Couleur d'arrière-plan du menu. Il s'agit du seul style de couleur qui n'hérite pas de sa valeur.
`color`	Couleur de l'étiquette de texte d'un élément de menu.
`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`.
`rollOverColor`	Couleur des éléments de menu survolés par le curseur.
`selectionColor`	Eléments sélectionnés et éléments contenant des sous-menus.

Style	Description
selectionDisabledColor	Eléments sélectionnés et éléments contenant des sous-menus et étant désactivés.
textRollOverColor	Couleur du texte lorsque le curseur passe sur un élément.
textDecoration	Décoration du texte : none ou underline.
textDisabledColor	Couleur du texte désactivé.
textSelectedColor	Couleur du texte d'un élément de menu sélectionné.
popupDuration	Durée de la transition lorsqu'un menu s'ouvre. La valeur 0 signifie aucune transition.

Utilisation d'enveloppes avec le composant Menu

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

Classe Menu (Flash Professionnel uniquement)

Héritage UIObject > UIComponent > View > ScrollView > ScrollSelectList > Menu

Nom de classe ActionScript mx.controls.Menu

Méthodes de la classe Menu

Méthode	Description
Menu.addMenuItem()	Ajoute un élément de menu au menu.
Menu.addMenuItemAt()	Ajoute un élément de menu à un endroit spécifique du menu.
Menu.createMenu()	Crée une occurrence de la classe Menu. C'est une méthode statique.
Menu.getMenuItemAt()	Obtient une référence à un élément de menu, à un emplacement spécifique.
Menu.hide()	Ferme un menu.
Menu.indexOf()	Renvoie l'index d'un élément de menu donné.
Menu.removeAll()	Supprime tous les éléments d'un menu.
Menu.removeMenuItemAt()	Supprime un élément de menu à un emplacement spécifique d'un menu.
Menu.setMenuItemEnabled()	Indique si un élément de menu est activé (true) ou non (false).
Menu.setMenuItemSelected()	Indique si un élément de menu est sélectionné (true) ou non (false).
Menu.show()	Ouvre un menu à un emplacement spécifique ou à son emplacement précédent.

Hérite de toutes les méthodes des classes *UIObject*, *UIComponent*, ScrollView et ScrollSelectList.

Propriétés de la classe Menu

Propriété	Description
`Menu.dataProvider`	Source de données d'un menu.

Hérite de toutes les propriétés des classes *UIObject*, *UIComponent*, ScrollView et ScrollSelectList.

Evénements de la classe Menu

Evénement	Description
`Menu.change`	Diffusé lorsqu'un utilisateur sélectionne un élément.
`Menu.menuHide`	Diffusé lorsqu'un menu se ferme.
`Menu.menuShow`	Diffusé lorsqu'un menu s'ouvre.
`Menu.rollOut`	Diffusé lorsque le pointeur cesse de survoler un élément.
`Menu.rollOver`	Diffusé lorsque le pointeur passe au-dessus d'un élément.

Hérite de tous les événements des classes *UIObject*, *UIComponent*, ScrollView et ScrollSelectList

Menu.addMenuItem()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

Usage 1 :

```
monMenu.addMenu(objetInit)
```

Usage 2 :

```
monMenu.addMenu(élémentMenuEnfant)
```

Paramètres

`objetInit` Objet contenant les propriétés d'initialisation des attributs d'un élément de menu. Pour plus d'informations, consultez *A propos des attributs XML d'élément de menu*, page 389.

`élémentMenuEnfant` Objet nœud XML.

Renvoie

Une référence au nœud XML ajouté.

Description

Méthode : dans l'usage 1, ajoute un élément de menu à la fin du menu. L'élément de menu est construit à partir des valeurs fournies dans le paramètre *`objetInit`*. Dans l'usage 2, ajoute un élément de menu (nœud XML prédéfini) sous la forme d'un objet XML à la fin du menu. Si le nœud ajouté existe déjà, il est supprimé de son emplacement précédent.

Exemple

Usage 1 : L'exemple suivant ajoute un élément de menu à un menu :

```
monMenu.addMenuItem({ label:"Elément 1", type:"radio", selected:false,
  enabled:true, instanceName:"élémentRadio1", groupName:"monGroupeRadio" } );
```

Usage 2 : L'exemple suivant déplace un nœud d'un menu vers la racine d'un autre menu :

```
monMenu.addMenuItem(monDeuxièmeMenu.getMenuItemAt(monDeuxièmeMenu, 3));
```

Menu.addMenuItemAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

Usage 1 :

```
monMenu.addMenuItemAt(index, objetInit)
```

Usage 2 :

```
monMenu.addMenuItemAt(index, élémentMenuEnfant)
```

Paramètres

index Entier indiquant la position (parmi les nœuds enfants) à laquelle l'élément est ajouté.

objetInit Objet contenant les propriétés d'initialisation des attributs d'un élément de menu. Pour plus d'informations, consultez *A propos des attributs XML d'élément de menu*, page 389.

élémentMenuEnfant Objet nœud XML.

Renvoie

Une référence au nœud XML ajouté.

Description

Méthode : dans l'usage 1, ajoute un élément de menu (nœud enfant) à l'emplacement indiqué dans le menu. L'élément de menu est construit à partir des valeurs fournies dans le paramètre *objetInit*. Dans l'usage 2, ajoute un élément de menu (nœud XML prédéfini) sous la forme d'un objet XML à un emplacement spécifié dans le menu. Si le nœud ajouté existe déjà, il est supprimé de son emplacement précédent.

Exemple

Usage 1 : L'exemple suivant ajoute un nouveau nœud en tant que deuxième enfant de la racine du menu :

```
monMenu.addMenuItemAt(1, { label:"Elément 1", instanceName:"élémentRadio1",
  type:"radio", selected:false, enabled:true, groupName:"monGroupeRadio" } );
```

Usage 2 : L'exemple suivant déplace un nœud d'un menu vers le quatrième enfant de la racine d'un autre menu :

```
monMenu.addMenuItemAt(3, monDeuxièmeMenu.getMenuItemAt(monDeuxièmeMenu, 3));
```

Menu.change

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
objetDécoute.change = function(objetEvt){
  // insérez votre code ici
}
monMenu.addEventListener("change", objetDécoute)
```

Description

Evénement : diffusé à l'ensemble des écouteurs enregistrés chaque fois qu'un utilisateur provoque un changement dans le menu.

Les composants V2 font appel à un modèle d'événement dispatcher/écouteur. Lorsqu'un composant Menu diffuse un événement `change`, l'événement est géré par une fonction (également appelée *gestionnaire*) associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`objetEvt`*) au gestionnaire. 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. L'objet événement de l'événement `Menu.change` possède les propriétés supplémentaires suivantes :

- `menuBar` Référence à l'occurrence de composant MenuBar qui est le parent du Menu cible. Elle prend la valeur undefined si le menu cible n'appartient pas à un menu.
- `menu` Référence à l'occurrence de composant Menu contenant l'élément cible.
- `menuItem` Nœud XML représentant l'élément de menu sélectionné.
- `groupName` Chaîne indiquant le nom du groupe de boutons radios auquel l'élément appartient. Elle prend la valeur undefined si l'élément n'appartient pas à un groupe de boutons radios.

Pour plus d'informations sur les objets événement, consultez *Objets événement*, page 592.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `écouteur` est défini et transmis à la méthode `monMenu.addEventListener()` en tant que deuxième paramètre. L'objet événement est capturé par le gestionnaire `change` du paramètre `événement`. Lorsque l'événement `change` est diffusé, une instruction `trace` est envoyée au panneau Sortie, comme suit :

```
listener = new Object();
écouteur.change = function(evt){
    trace("Elément de menu sélectionné : "+evt.menuItem.attributes.label);
}
monMenu.addEventListener("change", écouteur)
```

Menu.createMenu()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
Menu.createMenu(parent, mdp)
```

Paramètres

parent Occurrence du composant MovieClip. Le clip est le composant parent contenant la nouvelle occurrence de menu. Ce paramètre est facultatif.

mdp Occurrence de composant MenuDataProvider décrivant cette occurrence de menu. Ce paramètre est facultatif.

Renvoie

Une référence à la nouvelle occurrence de menu.

Description

Méthode (statique) : crée une occurrence de menu et l'associe (éventuellement) au parent indiqué ; la classe MenuDataProvider est spécifiée en tant que source de données des éléments de menu.

Si l'argument *parent* est omis ou null, le menu est associé au scénario `_root`.

Si l'argument *mdp* est omis ou null, le menu ne contient aucun élément ; vous devez appeler les méthodes `addMenu()` ou `setDataProvider()` pour l'alimenter.

Exemple

Dans l'exemple suivant, la ligne 1 crée une occurrence du composant MenuDataProvider, c'est-à-dire un objet XML décoré avec les méthodes de la classe MenuDataProvider. Les lignes suivantes ajoutent un élément de menu (Nouveau) et des sous-menus (Fichier, Projet et Ressource). Le bloc de code suivant ajoute d'autres éléments au menu principal. Le troisième bloc de code crée un menu vide associé à `monClipParent`, l'alimente avec la source de données `monMDP` (pour MenuData Provider) et l'ouvre à l'emplacement spécifié (coordonnées 100, 20), comme suit :

```
var monMDP = new XML();

var élémentNouveau = monMDP.addMenuItem({label:"Nouveau"});
    élémentNouveau.addMenuItem({label:"Fichier..."});
    élémentNouveau.addMenuItem({label:"Projet..."});
    élémentNouveau.addMenuItem({label:"Ressource..."});

monMDP.addMenuItem({label:"Ouvrir", instanceName:"miOuvrir"});
monMDP.addMenuItem({label:"Enregistrer", instanceName:"miEnregistrer"});
monMDP.addMenuItem({type:"separator"});
monMDP.addMenuItem({label:"Quitter", instanceName:"miQuitter"});

var monMenu = mx.controls.Menu.createMenu(monClipParent, monMDP);

monMenu.show(100, 20);
```

Pour tester ce code, placez-le sur l'image 1 du scénario principal dans le panneau Actions. Faites glisser un composant Menu du panneau Composants vers la scène et supprimez-le. Il est ajouté à la bibliothèque, mais pas au document.

Menu.dataProvider

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monMenu.dataProvider
```

Description

Propriété : la source de données des éléments d'un composant Menu.

`Menu.dataProvider` est un objet nœud XML. La définition de cette propriété remplace la source de données actuelle du menu.

La valeur par défaut est undefined.

Remarque : Toutes les occurrences XML ou XMLNode héritent automatiquement des méthodes et propriétés de l'API MenuDataProvider lorsqu'elles sont utilisées avec le composant Menu.

Exemple

L'exemple suivant importe un fichier XML et l'affecte à la propriété `Menu.dataProvider` :

```
var monMenuDP = new XML();
monMenuDP.load("http://monServeur.monDomaine.com/source.xml");
monMenuDP.onLoad = function(){
  maCommandeMenu.dataProvider = monMenuDP;
}
```

Menu.getMenuItemAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monMenu.getMenuItemAt(index)
```

Paramètres

index Entier spécifiant l'index du nœud dans le menu.

Renvoie

Une référence au nœud spécifié.

Description

Méthode : renvoie une référence au nœud enfant spécifié dans le menu.

Exemple

L'exemple suivant obtient une référence au deuxième nœud enfant de `monMenu` et en copie la valeur dans la variable `monElément` :

```
var monElément = monMenu.getMenuItemAt(1);
```

Menu.hide()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monMenu.hide()
```

Paramètres

index Index de l'élément Menu.

Renvoie

Rien.

Description

Méthode : ferme un menu en appliquant éventuellement des effets de transition.

Exemple

L'exemple suivant permet de réduire un menu développé :

```
monMenu.hide();
```

Voir aussi

`Menu.show()`

Menu.indexOf()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monMenu.indexOf(élément)
```

Paramètres

élément Référence à un nœud XML qui décrit un élément de menu.

Renvoie

L'index de l'élément de menu spécifié, ou undefined si l'élément n'appartient pas au menu.

Description

Méthode : renvoie l'index, dans cette occurrence de menu, de l'élément de menu spécifié.

Exemple

L'exemple suivant ajoute un élément de menu à un élément parent, puis obtient l'index de l'élément dans ce parent :

```
var monElément = monMenu.addMenuItem({label:"CetElément"});
var monIndex = monMenu.indexOf(monElément);
```

Menu.menuHide

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
objetDécoute.menuHide = function(objetEvt){
  // insérez votre code ici
}
monMenu.addEventListener("menuHide", objetDécoute)
```

Description

Evénement : diffusé à l'ensemble des écouteurs enregistrés chaque fois qu'un menu se ferme.

Les composants V2 font appel à un modèle d'événement dispatcher/écouteur. Lorsqu'un composant Menu distribue un événement `menuHide`, l'événement est géré par une fonction (également appelée *gestionnaire*) associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire et le nom de l'objet d'écoute en tant que paramètres.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`objetEvt`*) au gestionnaire. 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. L'objet événement de l'événement `Menu.menuHide` possède deux propriétés supplémentaires :

- `menuBar` Référence à l'occurrence de MenuBar qui est le parent du Menu cible. Elle prend la valeur undefined si le menu cible n'appartient pas à une occurrence de MenuBar.
- `menu` Référence à l'occurrence de Menu masquée.

Pour plus d'informations sur les objets événement, consultez *Objets événement*, page 592.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `formulaire` est défini et transmis à la méthode `monMenu.addEventListener()` en tant que deuxième paramètre. L'objet événement est capturé par le gestionnaire `menuHide` du paramètre `événement`. Lorsque l'événement `menuHide` est diffusé, une instruction `trace` est envoyée vers le panneau de sortie, comme suit :

```
form = new Object();
formulaire.menuHide = function(evt){
    trace("Menu fermé : "+evt.menu);
}
monMenu.addEventListener("menuHide", formulaire);
```

Voir aussi

`Menu.menuShow`

Menu.menuShow

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
objetDécoute.menuShow = function(objetEvt){
  // insérez votre code ici
}
monMenu.addEventListener("menuShow", objetDécoute)
```

Description

Evénement : diffusé à l'ensemble des écouteurs enregistrés chaque fois qu'un menu s'ouvre. Tous les nœuds parent ouvrent leurs menus pour afficher leurs enfants.

Les composants V2 font appel à un modèle d'événement dispatcher/écouteur. Lorsqu'un composant Menu diffuse un événement `menuShow`, l'événement est géré par une fonction (également appelée *gestionnaire*) associée à un objet d'écoute (`objetDécoute`) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire et le nom de l'objet d'écoute en tant que paramètres.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (`objetEvt`) au gestionnaire. 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. L'objet événement de l'événement `Menu.menuShow` possède les propriétés supplémentaires suivantes :

- `menuBar` Référence à l'occurrence de composant MenuBar qui est le parent du Menu cible. La valeur est undefined si le Menu cible n'appartient pas à un menu.
- `menu` Référence à l'occurrence du composant Menu affichée.

Pour plus d'informations sur les objets événement, consultez *Objets événement*, page 592.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `formulaire` est défini et transmis à la méthode `monMenu.addEventListener()` en tant que deuxième paramètre. L'objet événement est capturé par le gestionnaire `menuShow` du paramètre `objetEvt`. Lorsque l'événement `menuShow` est diffusé, une instruction `trace` est envoyée vers le panneau de sortie, comme suit :

```
form = new Object();
formulaire.menuShow = function(evt){
    trace("Menu ouvert : "+evt.menu);
}
monMenu.addEventListener("menuShow", formulaire);
```

Voir aussi

`Menu.menuHide`

Menu.removeAll()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monMenu.removeAll()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : supprime tous les éléments et actualise le menu.

Exemple

L'exemple suivant supprime tous les nœuds du menu :

```
monMenu.removeAll();
```

Menu.removeMenuItemAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monMenu.removeMenuItemAt(index)
```

Paramètres

index Index de l'élément de menu à supprimer.

Renvoie

Une référence à l'élément de menu renvoyé (nœud XML). La valeur est undefined s'il n'y a aucun élément à cette position.

Description

Méthode : supprime l'élément de menu et tous ses enfants à l'index spécifié. Si aucun élément de menu n'existe à l'index spécifié, l'appel de cette méthode n'a aucun effet.

Exemple

L'exemple suivant supprime un élément de menu à l'index 3 :

```
var Elément = monMenu.removeMenuItemAt(3);
```

Menu.rollOut

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
objetDécoute.rollOut = function(objetEvt){
  // insérez votre code ici
}
monMenu.addEventListener("rollOut", objetDécoute)
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque le pointeur cesse de survoler un élément de menu.

Les composants V2 font appel à un modèle d'événement dispatcher/écouteur. Lorsqu'un composant Menu diffuse un événement `rollOut`, l'événement est géré par une fonction (également appelée *gestionnaire*) associée à un objet d'écoute (*objetDécoute*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*objetEvt*) au gestionnaire. 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. L'objet événement de l'événement `Menu.rollOut` possède la propriété supplémentaire suivante :

- `élémentMenu` Référence à l'élément de menu (nœud XML) que le pointeur a cessé de survoler.

Pour plus d'informations sur les objets événement, consultez *Objets événement*, page 592.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `formulaire` est défini et transmis à la méthode `monMenu.addEventListener()` en tant que deuxième paramètre. L'objet événement est capturé par le gestionnaire `rollOut` du paramètre `événement`. Lorsque l'événement `rollOut` est diffusé, une instruction `trace` est envoyée vers le panneau de sortie, comme suit :

```
form = new Object();
formulaire.rollOut = function(evt){
    trace("Fin du survol du menu : "+evt.menuItem.attributes.label);
}
monMenu.addEventListener("rollOut", formulaire)
```

Menu.rollOver

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
objetDécoute.rollOver = function(objetEvt){
  // insérez votre code ici
}
monMenu.addEventListener("rollOver", objetDécoute)
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque le pointeur passe au-dessus d'un élément de menu.

Les composants V2 font appel à un modèle d'événement dispatcher/écouteur. Lorsqu'un composant Menu diffuse un événement `change`, l'événement est géré par une fonction (également appelée *gestionnaire*) associée à un objet d'écoute (*`objetDécoute`*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*`objetEvt`*) au gestionnaire. 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. L'objet événement de l'événement `Menu.rollOver` possède la propriété supplémentaire suivante :

`élémentMenu` Référence à l'élément de menu (nœud XML) que le pointeur a survolé.

Pour plus d'informations sur les objets événement, consultez *Objets événement*, page 592.

Exemple

Dans l'exemple suivant, un gestionnaire appelé `formulaire` est défini et transmis à la méthode `monMenu.addEventListener()` en tant que deuxième paramètre. L'objet événement est capturé par le gestionnaire `rollOver` du paramètre `événement`. Lorsque l'événement `rollOver` est diffusé, une instruction `trace` est envoyée vers le panneau de sortie, comme suit :

```
form = new Object();
formulaire.rollOver = function(evt){
    trace("Survol du menu : "+evt.menuItem.attributes.label);
}
monMenu.addEventListener("rollOver", formulaire)
```

Menu.setMenuItemEnabled()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monMenu.setMenuItemEnabled(élément, actif)
```

Paramètres

élément Nœud XML. Le nœud de l'élément de menu cible dans le fournisseur de données.

actif Valeur booléenne indiquant si l'élément est activé (`true`) ou non (`false`).

Renvoie

Rien.

Description

Méthode : applique à l'attribut `enabled` de l'élément cible l'état fourni par le paramètre *actif*. Si cet appel entraîne un changement d'état, l'élément est redessiné avec le nouvel état.

Exemple

L'exemple suivant désactive le deuxième enfant de `monMenu` :

```
var monElément = monMenu.getMenuItemAt(1);
monMenu.setMenuItemEnabled(monElément, false);
```

Voir aussi

`Menu.setMenuItemSelected()`

Menu.setMenuItemSelected()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monMenu.setMenuItemSelected(élément, sélectionné)
```

Paramètres

élément Nœud XML. Le nœud de l'élément de menu cible dans le fournisseur de données.

sélectionné Valeur booléenne indiquant si l'élément est sélectionné (`true`) ou non (`false`). Si l'élément est une case à cocher, elle peut être cochée selon sa valeur. Si l'élément est un bouton radio, il devient la sélection en cours dans le groupe radio.

Renvoie

Rien.

Description

Méthode : applique à l'attribut `selected` de l'élément l'état spécifié par le paramètre *sélectionné*. Si cet appel entraîne un changement d'état, l'élément est redessiné avec le nouvel état. Cette méthode n'a d'effet que sur les éléments dont l'attribut `type` est défini sur `"radio"` ou `"check"`, car elle modifie leur apparence. Si vous l'appelez sur un élément de `type "normal"` ou `"separator"`, aucun effet visible ne se produit.

Exemple

L'exemple suivant désélectionne le deuxième enfant de `monMenu` :

```
var monElément = monMenu.getMenuItemAt(1);
monMenu.setMenuItemSelected(monElément, false);
```

Menu.show()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monMenu.show(x, y)
```

Paramètres

x Coordonnée *x*.

y Coordonnée *y*.

Renvoie

Rien.

Description

Méthode : ouvre un menu à un emplacement spécifique. Le menu est automatiquement redimensionné pour que tous les éléments de premier niveau soient visibles et que son coin supérieur gauche soit placé à l'emplacement indiqué dans le système de coordonnées fourni par le parent du composant.

Si les paramètres *x* et *y* sont omis, le menu apparaît à son emplacement précédent.

Exemple

L'exemple suivant étend un menu :

```
monMenu.show(10, 10);
```

Voir aussi

`Menu.hide()`

Classe MenuDataProvider

La classe MenuDataProvider est une API Decorator (mix-in) qui complète la fonctionnalité de la classe globale XMLNode. Cette fonctionnalité permet aux occurrences XML affectées à une propriété Menu dataProvider de manipuler leurs propres données ainsi que les affichages de menu correspondants par l'intermédiaire de l'API MenuDataProvider.

Principaux concepts :

- MenuDataProvider est une API Decorator (mix-in). Elle n'a pas besoin d'être instanciée pour être utilisée.
- En mode natif, les menus acceptent la valeur XML pour la propriété `dataProvider`.
- Si une classe Menu est instanciée, toutes les occurrences XML du fichier SWF sont décorées par l'API MenuDataProvider.
- Seules les méthodes de l'API MenuDataProvider diffusent des événements aux commandes de menu. Vous pouvez toutefois utiliser des méthodes XML natives, mais celles-ci ne serviront pas d'événements de diffusion destinés à actualiser l'affichage des menus.
  - Utilisez les méthodes de l'API MenuDataProvider pour contrôler le modèle de données.
  - Utilisez les méthodes XML pour les opérations en lecture seule (parcourir la hiérarchie d'un menu, par exemple).
- Tous les éléments du menu sont des objets XML décorés à l'aide de l'API MenuDataProvider.
- Les modifications apportées aux attributs des éléments n'apparaissent pas dans le menu tant que celui-ci n'est pas redessiné.

Méthodes de la classe MenuDataProvider

Méthode	Description
`MenuDataProvider.addMenuItem()`	Ajoute un élément enfant.
`inboxMenuDataProvider.addMenuItemAt()`	Ajoute un élément enfant à un emplacement spécifique.
`MenuDataProvider.getMenuItemAt()`	Obtient une référence à un élément de menu, à un emplacement spécifique.
`MenuDataProvider.indexOf()`	Renvoie l'index d'un élément de menu donné.
`MenuDataProvider.removeMenuItem()`	Supprime un élément de menu.
`MenuDataProvider.removeMenuItemAt()`	Supprime un élément de menu à un endroit spécifique.

MenuDataProvider.addMenuItem()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

Usage 1 :

```
monMenu.addMenuItem(objetInit)
```

Usage 2 :

```
monMenu.addMenuItem(élémentMenuEnfant)
```

Paramètres

`objetInit` Objet contenant les attributs spécifiques qui initialisent les attributs d'un élément de menu. Pour plus d'informations, consultez *A propos des attributs XML d'élément de menu*, page 389.

`élémentMenuEnfant` Nœud XML.

Renvoie

Une référence à un objet XMLNode.

Description

Méthode : dans l'usage 1, ajoute un élément enfant à la fin d'un élément de menu parent (il peut s'agir du menu lui-même). L'élément de menu est construit à partir des valeurs transmises dans le paramètre *`objetInit`*. Dans l'usage 2, ajoute un élément de menu enfant défini dans le paramètre XML *`élémentMenuEnfant`* spécifié à la fin de l'élément de menu parent.

Exemple

L'exemple suivant ajoute un nouveau nœud au nœud spécifié dans le menu :

```
monMenuDP.firstChild.addMenuItem("Boîte de réception", { label:"Elément 1",
  icon:"icôneElémentRadio", type:"radio", selected:false, enabled:true,
  instanceName:"élémentRadio1", groupName:"monGroupeRadio" } );
```

inboxMenuDataProvider.addMenuItemAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

Usage 1 :

```
monMenu.addMenuItemAt(index, objetInit)
```

Usage 2 :

```
monMenu.addMenuItemAt(index, élémentMenuEnfant)
```

Paramètres

index Entier.

objetInit Objet contenant les attributs spécifiques qui initialisent les attributs d'un élément de menu. Pour plus d'informations, consultez *A propos des attributs XML d'élément de menu*, page 389.

élémentMenuEnfant Nœud XML.

Renvoie

Une référence au nœud XML ajouté.

Description

Méthode : dans l'usage 1, ajoute un élément enfant à la position d'index indiquée dans l'élément de menu parent (il peut s'agir du menu lui-même). L'élément de menu est construit à partir des valeurs transmises dans le paramètre *objetInit*. Dans l'usage 2, ajoute un élément de menu enfant défini dans le paramètre XML *élémentMenuEnfant* spécifié à l'index spécifié d'un élément de menu parent.

Exemple

Usage 1 : L'exemple suivant ajoute un nouveau nœud en tant que deuxième enfant de la racine du menu :

```
monMenu.addMenuItemAt(1, { label:"Elément 1", type:"radio", selected:false,
  enabled:true, instanceName:"elémentRadio1", groupName:"monGroupeRadio" } );
```

MenuDataProvider.getMenuItemAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monMenu.getMenuItemAt(index)
```

Paramètres

index Entier indiquant la position du menu.

Renvoie

Une référence au nœud XML spécifié.

Description

Méthode : renvoie une référence à l'élément de menu enfant spécifié dans l'élément de menu en cours.

Exemple

L'exemple suivant trouve le nœud que vous recherchez, puis obtient le deuxième enfant de `monElémentDeMenu` :

```
var monElémentDeMenu = monMenuDP.firstChild.firstChild;
monElémentDeMenu.getMenuItemAt(1);
```

MenuDataProvider.indexOf()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monMenu.indexOf(élément)
```

Paramètres

élément Référence au nœud XML qui décrit l'élément de menu.

Renvoie

L'index de l'élément de menu spécifié ; renvoie undefined si l'élément n'appartient pas au menu.

Description

Méthode : renvoie l'index de l'élément de menu spécifié dans l'élément de menu parent.

Exemple

L'exemple suivant ajoute un élément de menu à un élément parent, puis obtient l'index de l'élément :

```
var monElément = monElémentParent.addMenuItem({label:"CetElément"});
var monIndex = monElémentParent.indexOf(monElément);
```

MenuDataProvider.removeMenuItem()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monMenu.removeMenuItem()
```

Paramètres

Aucun.

Renvoie

Une référence à l'élément de menu supprimé (nœud XML) ; undefined si une erreur se produit.

Description

Méthode : supprime l'élément cible et ses nœuds enfant, le cas échéant.

Exemple

L'exemple suivant supprime `monElémentDeMenu` de son parent :

```
monElémentDeMenu.removeMenuItem();
```

MenuDataProvider.removeMenuItemAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monMenu.removeMenuItemAt(index)
```

Paramètres

index Index de l'élément Menu.

Renvoie

Une référence à l'élément de menu supprimé. La valeur est undefined s'il n'y a aucun élément à cette position.

Description

Méthode : supprime l'élément enfant de l'élément de menu spécifié par le paramètre *index*. Si aucun élément de menu n'existe à l'index spécifié, l'appel de cette méthode n'a aucun effet.

Exemple

L'exemple suivant supprime le quatrième élément :

```
monMenuDP.removeMenuItemAt(3);
```

Composant MenuBar (Flash Professionnel uniquement)

Le composant MenuBar permet de créer une barre de menus horizontale comportant des menus déroulants et des commandes, d'un type similaire à celui des barres de menu Fichier et Edition rencontrées dans les applications les plus courantes (comme Macromedia Flash). La barre de menus complète le composant Menu en fournissant une interface dans laquelle l'utilisateur peut, à l'aide de la souris ou du clavier, afficher ou masquer des menus qui se comportent comme un groupe.

La barre de menus permet de créer un menu d'application en quelques étapes seulement. Pour créer une barre de menus, vous pouvez soit affecter un fournisseur de données XML à la barre de menus pour décrire une série de menus, soit utiliser la méthode `MenuBar.addMenu()` pour ajouter des occurrences de menu de façon individuelle.

Dans la barre de menus, chaque menu se compose de deux parties : le menu et le bouton qui ouvre le menu (appelé activateur de menu). Les activateurs de menu sur lesquels vous pouvez cliquer apparaissent dans la barre de menus sous la forme d'étiquettes de texte comportant des bordures incrustées ou en relief, selon leur état de mise en valeur. Ces étiquettes réagissent aux interactions de la souris et du clavier.

Lorsque l'utilisateur clique sur l'activateur d'un menu, ce dernier s'ouvre sous l'activateur. Le menu reste actif jusqu'à ce que l'utilisateur clique à nouveau sur l'activateur, ou jusqu'à ce qu'il sélectionne un élément de menu ou clique hors de la zone du menu.

Outre les activateurs de menu, qui permettent d'afficher et de masquer les menus, la barre de menus regroupe plusieurs menus sous un même comportement. L'utilisateur peut ainsi parcourir un grand nombre de commandes en faisant glisser la souris sur les séries d'activateurs ou en utilisant les touches de direction pour passer d'une liste à l'autre. L'interactivité avec la souris est conjuguée à l'interactivité avec le clavier pour permettre à l'utilisateur de passer d'un menu à l'autre dans le composant MenuBar.

Un utilisateur ne peut pas parcourir les menus d'une barre de menus. Si des menus dépassent la largeur de la barre de menus, ils sont masqués.

Vous ne pouvez pas rendre le composant MenuBar accessible aux lecteurs d'écran.

Interaction avec le composant MenuBar (Flash Professionnel uniquement)

Vous pouvez utiliser la souris et le clavier pour interagir avec le composant MenuBar.

Lorsque vous passez la souris sur un activateur de menu, l'étiquette de l'activateur est mise en valeur par une bordure en relief.

Lorsqu'une occurrence de MenuBar a le focus (après que l'utilisateur a cliqué ou utilisé la touche de tabulation), il est possible d'utiliser les touches suivantes pour la contrôler :

Touche	Description
Flèche vers le bas	Déplace la sélection d'un élément plus bas dans le menu.
Flèche vers le haut	Déplace la sélection d'un élément plus haut dans le menu.

Touche	Description
Flèche vers la droite	Déplace la sélection vers le bouton suivant.
Flèche vers la gauche	Déplace la sélection vers le bouton précédent.
Entrée/Echap	Ferme un menu ouvert.

Remarque : Lorsqu'un menu est ouvert, vous ne pouvez pas utiliser la touche de tabulation pour le fermer. Vous devez soit faire une sélection, soit fermer le menu en appuyant sur Echap.

Utilisation du composant MenuBar (Flash Professionnel uniquement)

Vous pouvez utiliser le composant MenuBar pour ajouter un jeu de menus (par exemple, Fichier, Edition, Spécial, Fenêtre, etc.) à la bordure supérieure d'une application.

Paramètres du composant MenuBar

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

labels Tableau qui ajoute à la barre de menus des activateurs de menu portant les étiquettes indiquées. La valeur par défaut est [] (tableau vide).

Vous pouvez contrôler ces options et d'autres options du composant MenuBar à l'aide des propriétés, méthodes et événements d'ActionScript. Pour plus d'informations, consultez *Classe MenuBar*, page 418.

Création d'une application avec le composant MenuBar

Dans cet exemple, vous faites glisser un composant MenuBar vers la scène, ajoutez du code pour remplir l'occurrence avec des menus et associez à ces menus des écouteurs qui réagiront aux sélections de l'utilisateur.

Pour utiliser un composant MenuBar dans une application :

1 Choisissez Fichier > Nouveau pour créer un document Flash.

2 Faites glisser le composant MenuBar du panneau Composants jusqu'à la scène.

3 Placez le menu en haut de la scène pour obtenir une disposition standard.

4 Sélectionnez la barre de menus et saisissez le nom d'occurrence **maBarreDeMenus** dans l'inspecteur des propriétés.

5 Dans le panneau Actions, sur l'Image 1, entrez le code suivant :

```
var menu = maBarreDeMenus.addMenu("Fichier");
menu.addMenuItem({label:"Nouveau", instanceName:"occurrenceNouveau"});
menu.addMenuItem({label:"Ouvrir", instanceName:"occurrenceOuvrir"});
menu.addMenuItem({label:"Fermer", instanceName:"occurrenceFermer"});
```

Ce code ajoute un menu Fichier à l'occurrence de barre de menus. Il utilise ensuite l'API Menu pour ajouter trois éléments de menu : Nouveau, Ouvrir et Fermer.

6 Dans le panneau Actions, sur l'Image 1, entrez le code suivant :

```
var écouter = new Object();
écouter.change = function(evt){
  var menu = evt.menu;
  var item = evt.menuItem
  if (item == menu.occurrenceNouveau){
    monNouveau();
    trace(élément);
  }else if (item == menu.occurrenceOuvrir){
    monOuvrir()
    trace(élément);
  }
}
menu.addEventListener("change",écouter);
```

Ce code crée un objet d'écoute, `écouter`, qui utilise l'objet événement `evt` pour capturer les sélections d'élément de menu.

Remarque : Vous devez appeler la méthode `addEventListener` pour enregistrer l'écouteur avec l'occurrence de menu et non avec l'occurrence de barre de menus.

7 Choisissez Contrôle > Tester l'animation pour tester le composant MenuBar.

Personnalisation du composant MenuBar (Flash Professionnel uniquement)

La taille de ce composant s'ajuste automatiquement en fonction des étiquettes d'activateur fournies par la propriété `dataProvider` ou les méthodes de la classe MenuBar. Lorsqu'un bouton d'activateur se trouve sur une barre de menus, il conserve une taille fixe qui est fonction des styles de police et de la longueur du texte.

Utilisation de styles avec le composant MenuBar

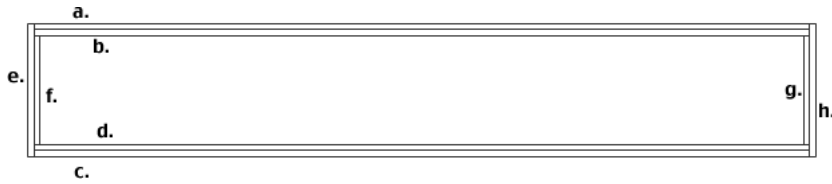
Le composant MenuBar crée une étiquette d'activateur pour chaque menu à l'intérieur d'un groupe. Vous pouvez utiliser des styles pour modifier l'aspect des étiquettes d'activateur. Un composant MenuBar supporte les styles de halo suivants :

Style	Description
`themeColor`	La couleur de mise en valeur d'une sélection. 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".
`popupDuration`	Le temps en millisecondes qui s'écoule avant qu'un menu s'ouvre. La valeur par défaut est 0.

Le composant MenuBar utilise également la classe RectBorder pour dessiner les bordures de mise en valeur incrustée et en relief reflétant les interactions utilisateur avec les différents éléments. 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 :



Utilisation d'enveloppes avec le composant MenuBar

Le composant MenuBar utilise les enveloppes du composant Menu pour représenter ses états visuels. Pour plus d'informations sur les enveloppes du composant Menu, consultez *Utilisation d'enveloppes avec le composant Menu*, page 396.

Classe MenuBar

Héritage UIObject > UIComponent > MenuBar

Nom de classe ActionScript mx.controls.MenuBar

Méthodes de la classe MenuBar

Méthode	Description
`MenuBar.addMenu()`	Ajoute un menu à la barre de menus.
`MenuBar.addMenuAt()`	Ajoute un menu à un emplacement spécifique de la barre de menus.
`MenuBar.getMenuAt()`	Obtient une référence à un menu, à un emplacement spécifique.
`MenuBar.getMenuEnabledAt()`	Renvoie une valeur booléenne indiquant si un menu est activé (`true`) ou non (`false`).

Méthode	Description
`MenuBar.removeMenuAt()`	Supprime un menu à un emplacement spécifique d'une barre de menus.
`MenuBar.setMenuEnabledAt()`	Valeur booléenne indiquant si un menu est activé (`true`) ou non (`false`).

Hérite de toutes les méthodes des classes *UIObject* et *UIComponent*.

Propriétés de la classe MenuBar

Propriété	Description
`MenuBar.dataProvider`	Modèle de données d'une barre de menus.
`MenuBar.labelField`	Chaîne déterminant l'attribut de chaque objet XMLNode à utiliser en tant que texte d'étiquette de l'élément de barre de menus.
`MenuBar.labelFunction`	Fonction déterminant le texte à afficher dans l'étiquette de chaque élément de la barre de menus.

Hérite de toutes les méthodes des classes *UIObject* et *UIComponent*.

MenuBar.addMenu()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

Usage 1 :

```
maBarreDeMenus.addMenu(étiquette)
```

Usage 2 :

```
maBarreDeMenus.addMenu(étiquette, menuDataProvider)
```

Paramètres

`étiquette` Chaîne indiquant l'étiquette du nouveau menu.

`menuDataProvider` Occurrence XML ou XMLNode décrivant le menu et ses éléments. S'il s'agit d'une occurrence XML, le premier enfant (`firstChild`) de l'occurrence est utilisé.

Renvoie

Une référence au nouvel objet Menu.

Description

Méthode : dans l'usage 1, ajoute un seul menu et un seul activateur de menu à la fin de la barre de menus avec la valeur spécifiée dans le paramètre *`étiquette`*. Dans l'usage 2, ajoute un seul menu et un seul activateur de menu, définis dans le paramètre XML *`menuDataProvider`* spécifié.

Exemple

Usage 1 : L'exemple suivant ajoute un menu Fichier, puis utilise la méthode `Menu.addMenuItem()` pour ajouter les éléments de menu Nouveau et Ouvrir :

```
menu = maBarreDeMenus.addMenu("Fichier");
menu.addMenuItem({label:"Nouveau", instanceName:"occurrenceNouveau"});
menu.addMenuItem("{label:"Ouvrir", instanceName:"occurrenceOuvrir"}");
```

Usage 2 : L'exemple suivant ajoute un menu Police contenant les éléments de menu Gras et Italique définis dans le paramètre `monMenuDP2` de menuDataProvider :

```
var monMenuDP2 = new XML();
monMenuDP2.addMenuItem({type:"check", label:"Gras", instanceName:"cocher1"});
monMenuDP2.addMenuItem({type:"check", label:"Italique",
  instanceName:"cocher2"});
menu = maBarreDeMenus.addMenu("Police",monMenuDP2);
```

MenuBar.addMenuAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

Usage 1 :

```
maBarreDeMenus.addMenuAt(index, étiquette)
```

Usage 2 :

```
maBarreDeMenus.addMenuAt(index, étiquette, menuDataProvider)
```

Paramètres

`index` Entier indiquant la position à laquelle le menu doit être inséré. La première position est 0. Pour insérer le menu à la fin, appelez la méthode `MenuBar.addMenu(étiquette)`.

`étiquette` Chaîne indiquant l'étiquette du nouveau menu.

`menuDataProvider` Occurrence XML ou XMLNode décrivant le menu. S'il s'agit d'une occurrence XML, le premier enfant (`firstChild`) de l'occurrence est utilisé.

Renvoie

Une référence au nouvel objet Menu.

Description

Méthode : dans l'usage 1, ajoute un seul menu et un seul activateur de menu à l'*`index`* spécifié avec la valeur spécifiée dans le paramètre *`étiquette`*. Dans l'usage 2, ajoute un seul menu et un seul activateur de menu étiqueté à l'index spécifié. Le contenu du menu est défini dans le paramètre *`menuDataProvider`*.

Exemple

Usage 1 : L'exemple suivant place un menu à gauche de tous les menus du composant MenuBar :

```
menu = maBarreDeMenus.addMenuAt(0,"Toréador");
menu.addMenuItem("A propos de Macromedia Flash", instanceName:"occurApropos");
menu.addMenuItem("Préférences", instanceName:"occurPréf");
```

Usage 2 : L'exemple suivant ajoute un menu Edition comportant les éléments de menu Annuler, Répéter, Couper et Copier, définis dans le paramètre `monMenuDP` de *`menuDataProvider`* :

```
var monMenuDP = new XML();
monMenuDP.addMenuItem({label:"Annuler", instanceName:"occurAnnuler"});
monMenuDP.addMenuItem({label:"Répéter", instanceName:"occurRépéter"});
monMenuDP.addMenuItem({type:"separator"});
monMenuDP.addMenuItem({label:"Couper", instanceName:"occurCouper"});
monMenuDP.addMenuItem({label:"Copier", instanceName:"occurCopier"});

maBarreDeMenus.addMenuAt(0,"Edition",monMenuDP);
```

MenuBar.dataProvider

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maBarreDeMenus.dataProvider
```

Description

Propriété : le modèle de données des éléments d'un composant MenuBar.

`menuBar.dataProvider` est un objet nœud XML. La définition de cette propriété remplace le modèle de données actuel du composant MenuBar. Si le fournisseur de données possède des nœuds enfant, ceux-ci deviennent les éléments de la barre de menus ; les sous-nœuds de ces nœuds enfant deviennent les éléments de leurs menus respectifs.

La valeur par défaut est undefined.

Remarque : Toutes les occurrences XML ou XMLNode héritent automatiquement des méthodes et propriétés de l'API MenuDataProvider lorsqu'elles sont utilisées avec le composant MenuBar.

Exemple

L'exemple suivant importe un fichier XML et l'affecte à la propriété `MenuBar.dataProvider` :

```
var maBarreDeMenusDP = new XML();
maBarreDeMenusDP.load("http://monServeur.monDomaine.com/source.xml");
maBarreDeMenusDP.onLoad = function(succès){
  if(succès){
    maBarreDeMenus.dataProvider = maBarreDeMenusDP;
} else {
  trace("erreur lors du chargement du fichier XML");
  }
}
```

MenuBar.getMenuAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maBarreDeMenus.getMenuAt(index)
```

Paramètres

index Entier indiquant la position du menu.

Renvoie

Une référence au menu à l'index spécifié. La valeur est undefined s'il n'y a aucun menu à cette position.

Description

Méthode : renvoie une référence au menu à l'index spécifié.

Exemple

Etant donné que la méthode `getMenuAt()` renvoie une occurrence, vous pouvez ajouter des éléments dans un menu à l'index spécifié. Dans l'exemple suivant, une fois les activateurs de menu Fichier, Edition et Affichage créés à l'aide du paramètre Label, le code suivant permet d'ajouter les éléments Nouveau et Ouvrir au menu Fichier à l'exécution :

```
menu = maBarreDeMenus.getMenuAt(0);
menu.addMenuItem({label:"Nouveau", instanceName:"occurNouveau"});
menu.addMenuItem({label:"Ouvrir", instanceName:"occurOuvrir"});
```

MenuBar.getMenuEnabledAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maBarreDeMenus.getMenuEnabledAt(index)
```

Paramètres

index Index de l'élément MenuBar.

Renvoie

Une valeur booléenne indiquant si le menu peut être choisi (`true`) ou non (`false`).

Description

Méthode : renvoie une valeur booléenne indiquant si le menu peut être choisi (`true`) ou non (`false`).

Exemple

L'exemple suivant appelle la méthode sur le menu situé à la première position de `maBarreDeMenus` :

```
maBarreDeMenus.getMenuEnabledAt(0);
```

MenuBar.labelField

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maBarreDeMenus.labelField
```

Description

Propriété : une chaîne déterminant l'attribut de chaque XMLNode à utiliser en tant que texte d'étiquette du menu. Cette propriété est également transmise à tous les menus créés à partir de la barre de menus. La valeur par défaut est `"label"`.

Une fois la propriété `dataProvider` définie, cette propriété est en lecture seule.

Exemple

L'exemple suivant utilise l'attribut `nom` de chaque nœud en tant que texte d'étiquette :

```
maBarreDeMenus.labelField = "nom";
```

MenuBar.labelFunction

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maBarreDeMenus.labelFunction
```

Description

Propriété : une fonction déterminant le texte à afficher dans chaque étiquette de menu. Cette fonction accepte le nœud XML associé à l'élément en tant que paramètre et renvoie une chaîne à utiliser en tant que texte d'étiquette. Cette propriété est transmise à tous les menus créés à partir de la barre de menus. La valeur par défaut est undefined.

Une fois la propriété `dataProvider` définie, cette propriété est en lecture seule.

Exemple

L'exemple suivant crée une étiquette personnalisée à partir des attributs de nœud :

```
maBarreDeMenus.labelFunction = function(nœud){
var a = nœud.attributes;
return "Le prix de " + a.name + " est " + a.price;
};
```

MenuBar.removeMenuAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maBarreDeMenus.removeMenuAt(index)
```

Paramètres

index Index de l'élément MenuBar.

Renvoie

Une référence à l'élément MenuBar renvoyé. La valeur est undefined s'il n'y a aucun élément à cette position.

Description

Méthode : supprime l'élément de menu à l'index spécifié. Si aucun élément de menu n'existe à l'index spécifié, l'appel de cette méthode n'a aucun effet.

Exemple

L'exemple suivant supprime le menu à l'index 4 :

```
maBarreDeMenus.removeMenuAt(4);
```

MenuBar.setMenuEnabledAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
maBarreDeMenus.setMenuEnabledAt(index, booléen)
```

Paramètres

index L'index de l'élément MenuBar à définir.

booléen Une valeur booléenne indiquant si l'élément de menu à l'index spécifié est activé (`true`) ou non (`false`).

Renvoie

Rien.

Description

Méthode : active l'élément de menu à l'index spécifié. Si aucun menu n'existe à l'index spécifié, l'appel de cette méthode n'a aucun effet.

Exemple

L'exemple suivant obtient l'objet MenuBarColumn à l'index 3 :

```
maBarreDeMenus.setMenuEnabledAt(3);
```

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 27 ou *Classe FocusManager*, page 287.

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.

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 demi-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 :

```
classementFilm = new Object();
classementFilm.change = function (objetEvt){
  classementEtoiles.value = objetEvt.target.value;
}
stepperEtoile.addEventListener("change", classementFilm);
```

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 programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez la méthode `setSize()` (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 29.

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 38.

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 27.

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 méthodes des classes *UIObject* et *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 des classes *UIObject* et *UIComponent*.

Evénements de la classe NumericStepper

Evénement	Description
`NumericStepper.change`	Déclenché lorsque la valeur change.

Hérite de tous les événements des classes *UIObject* et *UIComponent*.

NumericStepper.change

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.change = function(objetEvt){
   ...
}
OccurrenceDeStepper.addEventListener("change", objetDécoute)
```

Description

Evénement : diffusé 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 (également appelée un *gestionnaire*), sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

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 `objEvt`, 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(objEvt){
  // objEvt.target est le composant ayant généré l'événement change,
  //c'est-à-dire, l'incrémenteur numérique.
  trace("Valeur passée à " + objEvt.target.value);
}
monIncrémenteurNumérique.addEventListener("change", form);
```

NumericStepper.maximum

Disponibilité

Flash Player 6 version 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;
```

Voir aussi

`NumericStepper.minimum`

NumericStepper.minimum

Disponibilité

Flash Player 6 version 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;
```

Voir aussi

NumericStepper.maximum

NumericStepper.nextValue

Disponibilité

Flash Player 6 version 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);
```

Voir aussi

NumericStepper.previousValue

NumericStepper.previousValue

Disponibilité

Flash Player 6 version 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);
```

Voir aussi

NumericStepper.nextValue

NumericStepper.stepSize

Disponibilité

Flash Player 6 version 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 version 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

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

Méthode	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 version 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 version 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 progression 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 progression 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 progression vous permet d'afficher la progression du contenu pendant le chargement. Ces informations sont essentielles pour l'utilisateur lorsqu'il interagit avec une application.

Vous pouvez utiliser le composant ProgressBar dans plusieurs modes. Définissez le mode à l'aide du paramètre mode. Les modes les plus 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 `getBytesTotal` (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 progression. Cette valeur peut être : event, polled ou manual. La valeur par défaut est event.

source Chaîne devant être convertie en objet représentant le nom d'occurrence de la source.

direction La direction dans laquelle avance la barre de progression. Cette valeur peut être right ou left, la valeur par défaut étant right.

label Texte indiquant la progression du chargement. Ce paramètre est une chaîne au format « %1 sur %2 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 progression. 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 progression est utilisée en mode event. En mode event, le contenu en cours de chargement doit émettre des événements `progress` et `complete` pour permettre à la barre de progression d'afficher la progression. Le composant Loader émet ces événements. Pour plus d'informations, consultez *Composant Loader*, page 334.

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 progression 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.contentPath = "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 progression est utilisée en mode polled. En mode polled, la barre de progression utilise les méthodes `getBytesLoaded` et `getBytesTotal` de l'objet source pour afficher la progression.

Pour créer une application avec le composant ProgressBar en mode polled, 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 progression 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 progression 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 programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez `UIObject.setSize()`.

Les embouts gauche et droit de la barre de progression et le graphique de suivi ont une taille fixe. Lorsque vous redimensionnez une barre de progression, sa partie centrale est redimensionnée pour s'ajuster entre les embouts. Si une barre de progression 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 progression. Si le nom d'une propriété de style se termine par « Color », il s'agit d'une propriété de style de couleur qui se comporte différemment des autres propriétés de style. Pour plus d'informations, consultez *Utilisation des styles pour personnaliser la couleur et le texte des composants*, page 29.

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 "italic".
`fontWeight`	Epaisseur de la police : "normal" ou "bold".
`textDecoration`	Décoration du texte : "none" ou "underline".

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 progression 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 38.

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 progression.

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`.
`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

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 progression se remplit.
`ProgressBar.indeterminate`	Indique que le nombre total d'octets de la source est inconnu.
`ProgressBar.label`	Texte accompagnant la barre de progression.
`ProgressBar.labelPlacement`	Emplacement de l'étiquette par rapport à la barre de progression.
`ProgressBar.maximum`	Valeur maximum de la barre de progression en mode manuel.
`ProgressBar.minimum`	Valeur minimum de la barre de progression en mode manuel.
`ProgressBar.mode`	Mode dans lequel la barre de progression charge le contenu.
`ProgressBar.percentComplete`	Nombre indiquant le pourcentage chargé.
`ProgressBar.source`	Contenu à charger dont la progression est contrôlée par la barre de progression.
`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 *UIObject*.

ProgressBar.complete

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(complete){
  ...
}
```

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.complete = function(objetEvt){
  ...
}
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 : diffusé 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 (*pBar*) distribue un événement (ici, `complete`) qui est géré par une fonction (également appelée *gestionnaire*) sur 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 transmet automatiquement un objet événement (*objetEvt*) 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 événement, consultez *Objets événement*, page 592.

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(objEvt){
  // objEvt.target est le composant ayant généré l'événement change,
  // c.-à-d., la barre de progression.
  trace("Valeur passée à " + objEvt.target.value);
}
barreP.addEventListener("complete", form);
```

Voir aussi

`UIEventDispatcher.addEventListener()`

ProgressBar.conversion

Disponibilité

Flash Player 6 version 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 réduit au minimum 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 version 79.

Edition

Flash MX 2004.

Usage

occurrenceBarreP.direction

Description

Propriété : indique la direction de remplissage de la barre de progression. La valeur par défaut est `"right"`.

Exemple

Le code suivant indique à la barre de progression de se remplir de droite à gauche.

```
barreP.direction = "left";
```

ProgressBar.indeterminate

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

```
occurrenceBarreP.indeterminate
```

Description

Propriété : valeur booléenne indiquant si la barre de progression a un remplissage rayé et une source de chargement de taille inconnue (`true`), ou un remplissage uni et une source de chargement de taille connue (`false`).

Exemple

Le code suivant crée une barre de progression 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 version 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 progression au format « 4 fichiers chargés » :

```
barreP.label = "%1 fichiers chargés";
```

Voir aussi

ProgressBar.labelPlacement

ProgressBar.labelPlacement

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

occurrenceBarreP.labelPlacement

Description

Propriété : définit le placement de l'étiquette par rapport à la barre de progression. Les valeurs possibles sont `"left"`, `"right"`, `"top"`, `"bottom"` et `"center"`.

Exemple

Le code suivant définit l'étiquette à afficher au-dessus de la barre de progression :

```
barreP.label = "%1 sur %2 chargés (%3%%)";
barreP.labelPlacement = "top";
```

Voir aussi

ProgressBar.label

ProgressBar.maximum

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

occurrenceBarreP.maximum

Description

Propriété : la plus grande valeur de la barre de progression 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;
```

Voir aussi

ProgressBar.minimum, ProgressBar.mode

ProgressBar.minimum

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

occurrenceBarreP.minimum

Description

Propriété : la plus petite valeur de la barre de progression lorsque la propriété ProgressBar.mode est définie sur "manual".

Exemple

Le code suivant définit la valeur minimum de la barre de progression :

```
barreP.minimum = 0;
```

Voir aussi

ProgressBar.maximum, ProgressBar.mode

ProgressBar.mode

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

occurrenceBarreP.mode

Description

Propriété : mode dans lequel la barre de progression 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 `getBytesTotal`, 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 progression en mode event :

```
barreP.mode = "event";
```

ProgressBar.percentComplete

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

`occurrenceBarreP``.percentComplete`

Description

Propriété (lecture seule) : renvoie le pourcentage de progression du processus. Cette valeur est réduite au minimum. 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 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(progress){
  ...
}
```

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.progress = function(objetEvt){
  ...
}
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 progression 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 (également appelée *gestionnaire*) sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

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(objEvt){
  // objEvt.target est le composant ayant généré l'événement change,
  // c.-à-d., la barre de progression.
  trace("Valeur passée à " + objEvt.target.value);
}
barreP.addEventListener("progress", form);
```

Voir aussi

`UIEventDispatcher.addEventListener()`

ProgressBar.setProgress()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

`occurrenceBarreP``.setProgress(`*`completed, total`*`)`

Paramètres

`completed` nombre indiquant le niveau de progression atteint. Vous pouvez utiliser les propriétés `ProgressBar.label` et `ProgressBar.conversion` pour afficher le nombre sous forme de pourcentage ou dans l'unité de votre choix, en fonction de la source de la barre de progression.

`total` nombre indiquant la progression totale devant être effectuée pour atteindre 100 pourcent.

Renvoie

Un 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 version 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 progression 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'occurrence correspond à `loader` :

```
barreP.source = loader;
```

Voir aussi

`ProgressBar.mode`

ProgressBar.value

Disponibilité

Flash Player 6 version 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. Seul un membre peut être sélectionné au sein du groupe. La sélection d'un bouton radio dans un groupe désélectionne le bouton jusqu'alors sélectionné dans le groupe. Vous 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 27 ou *Classe FocusManager*, page 287.

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 452.

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.

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 : left, right, top ou bottom ; la valeur par défaut est right. 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 programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez la méthode `setSize()` (consultez *UIObject.setSize()*, page 606).

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 29.

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 38.

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 38. 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 27.

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 des classes *UIObject*, *UIComponent*, SimpleButton et *Classe Button*.

Propriétés de la classe RadioButton

Propriété	Description
`RadioButton.data`	La valeur associée à une occurrence de bouton radio.
`RadioButton.groupName`	Le nom 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 des classes *UIObject*, *UIComponent*, SimpleButton et *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 des classes *UIObject*, *UIComponent*, SimpleButton et *Classe Button*.

RadioButton.click

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.click = function(objetEvt){
  ...
}
GroupeDeBoutonsRadio.addEventListener("click", objetDécoute)
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque l'utilisateur clique à l'aide de la souris (bouton de la souris enfoncé et relâché) sur le bouton radio ou si le bouton radio est sélectionné en utilisant les boutons fléchés. L'événement est 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 que 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 (également appelée *gestionnaire*), sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

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 `objEvt`, 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(objEvt){
  trace("L'occurrence radio sélectionnée est" + objEvt.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 version 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 version 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 version 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 version 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 version 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 version 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 version 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 (Flash Professionnel uniquement)

Les composants Resolver s'utilisent en association avec le composant DataSet (partie de la fonctionnalité de gestion des données dans l'architecture de données Macromedia Flash). Les composants Resolver vous permettent de convertir les modifications apportées aux données de votre application dans un format adapté à la source de données externe que vous mettez à jour. Ils ne sont pas visibles à l'exécution.

Si vous utilisez un composant DataSet dans votre application, il génère un jeu optimisé d'instructions (DeltaPacket) qui décrivent les modifications apportées aux données à l'exécution. Ce jeu d'instructions est converti au format approprié (paquet de mise à jour) par les composants Resolver. Lorsqu'une mise à jour est envoyée au serveur, celui-ci envoie une réponse (paquet de résultats) contenant d'autres mises à jour ou des erreurs résultant de l'opération de mise à jour. Les composants Resolver peuvent reconvertir ces informations en DeltaPacket pour l'appliquer au composant DataSet afin que ce dernier reste synchronisé avec la source de données externe. Les composants Resolver vous permettent de maintenir votre application synchronisée avec une source de données externe sans écrire de code ActionScript supplémentaire.

Le composant RDBMSResolver traduit le paquet XML qui peut être reçu et analysé par un service web, JavaBean, servlet ou une page ASP. Le paquet XML contient les informations et le formatage requis pour mettre à jour des bases de données relationnelles SQL standard. Le composant Resolver parallèle XUpdateResolver (voir *Composant XUpdateResolver (Flash Professionnel uniquement)*, page 665) permet de renvoyer les données à un serveur XML. Pour plus d'informations sur les composants DataSet, consultez *Composant DataSet (Flash Professionnel uniquement)*, page 207. Pour plus d'informations sur les connecteurs, consultez *WebServiceConnector (Flash Professionnel uniquement)*, page 636 et *Composant XMLConnector (Flash Professionnel uniquement)*, page 657. Pour plus d'informations sur l'architecture de données Flash, consultez « Composants Resolver (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Le composant RDBMSResolver convertit les modifications apportées aux données de votre application en un paquet XML, qui peut ensuite être envoyé à une source de données externe.

Remarque : Vous pouvez utiliser le composant RDBMSResolver pour envoyer des mises à jour de données à une source de données externe pouvant analyser les données XML et générer des instructions SQL destinées à une base de données (par exemple, une page ASP, un servlet Java ou un composant ColdFusion).

Les mises à jour issues du composant RDBMSResolver sont envoyées sous la forme d'un paquet de mise à jour XML, transmis à la base de données via un objet connecteur. Le composant Resolver est connecté à la propriété deltaPacket d'un composant DataSet ; il envoie son propre paquet de mise à jour à un connecteur, qui lui renvoie les erreurs serveur ; le composant Resolver transmet ces erreurs au composant DataSet. Tous ces processus utilisent les propriétés de liaison.

Utilisation du composant RDBMSResolver (Flash Professionnel uniquement)

Utilisez ce composant RDBMSResolver uniquement lorsque votre application Flash contient un composant DataSet et doit renvoyer une mise à jour à la source de données. Ce composant traduit les données que vous voulez renvoyer à une base de données relationnelles.

Pour plus d'informations sur l'utilisation du composant RDBMSResolver, consultez « Composants Resolver (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Paramètres du composant RDBMSResolver

TableName Chaîne représentant le nom de la table de base de données devant être mise à jour dans le paquet XML. Cette valeur doit être identique à celle saisie pour la mise à jour de l'élément `Resolver.fieldInfo`. S'il n'existe aucune mise à jour de ce champ, cette valeur doit rester vide (c'est la valeur par défaut).

UpdateMode Enumérateur déterminant la manière dont les champs-clés sont identifiés au moment de la génération du paquet de mise à jour XML. La valeur par défaut est `umUsingKey`. Les valeurs possibles sont :

- `umUsingAll` Utilise les anciennes valeurs de tous les champs modifiés pour identifier l'enregistrement à mettre à jour. Il s'agit de la valeur la plus sûre pour la mise à jour, car elle garantit que l'enregistrement n'a pas été modifié par un autre utilisateur depuis que vous l'avez récupéré. Toutefois, elle prend du temps et génère un paquet de mise à jour plus volumineux.
- `umUsingModified` Utilise les anciennes valeurs de tous les champs modifiés pour identifier l'enregistrement à mettre à jour. Cette valeur garantit qu'aucun autre utilisateur n'a modifié les mêmes champs de l'enregistrement depuis que vous l'avez récupéré.
- `umUsingKey` Il s'agit de la valeur par défaut. Ce paramètre utilise les anciennes valeurs des champs-clés. Il implique un « modèle de contrôle des accès concurrents optimiste », utilisé aujourd'hui par la plupart des systèmes de base de données, et garantit que vous modifiez l'enregistrement que vous avez récupéré de la base de données. Vos modifications remplacent celles qu'un autre utilisateur a apportées aux mêmes données.

NullValue Chaîne représentant une valeur de champ null. Vous pouvez la personnaliser pour éviter de la confondre avec une chaîne vide ("") ou une autre valeur valide. La valeur par défaut est `{_NULL_}`.

FieldInfo Collection représentant un ou plusieurs champs-clés qui identifient les enregistrements de façon unique. Si votre source de données est une table de base de données, alors elle possède un ou plusieurs champs qui identifient ses enregistrements de façon unique. En outre, certains champs peuvent avoir été calculés ou joints à partir d'une autre table. Ces champs doivent être identifiés pour que les champs-clés puissent être définis dans le paquet de mise à jour XML et pour que les champs qui ne doivent pas être mis à jour soient omis du paquet de mise à jour XML.

Le composant `RDBMSResolver` contient une propriété `FieldInfo` à cette fin. Cette propriété de collection vous permet de définir un nombre illimité de champs avec des propriétés identifiant les champs qui nécessitent un traitement particulier. Chaque élément `FieldInfo` de la collection contient trois propriétés :

- `FieldName` Nom d'un champ. Il doit correspondre à un champ du composant DataSet.
- `OwnerName` Valeur facultative utilisée pour identifier des champs qui ne sont pas inclus dans la table définie dans le paramètre `TableName` du composant Resolver. Si vous entrez la même valeur que celle du paramètre `TableName` ou le laissez vide, le champ est généralement inclus dans le paquet de mise à jour XML. Si vous entrez une autre valeur, le champ est exclu du paquet de mise à jour.
- `IsKey` Propriété booléenne que vous devez définir sur `true` pour que tous les champs-clés de la table soient mis à jour.

L'exemple suivant montre les éléments `FieldInfo` créés pour mettre à jour les champs de la table client. Vous devez identifier les champs-clés de la table client. La table client possède un champ-clé unique, `"id"` ; vous devez donc créer un élément de champ avec les valeurs suivantes :

```
FieldName = "id"
OwnerName = <--! laissez cette valeur vide -->
IsKey = "true"
```

Le champ `typeClient` est également ajouté à l'aide d'une jointure dans la requête. Ce champ doit être exclu de la mise à jour. Vous créez donc un élément de champ avec les valeurs suivantes :

```
FieldName = "typeClient"
OwnerName = "ChampJoint"
IsKey = "false"
```

Une fois les éléments de champ définis, Flash Player peut les utiliser pour générer automatiquement le paquet de mise à jour XML qui servira à mettre à jour une table.

Propriétés du composant RDBMSResolver

Propriété	Description
`RDBMSResolver.deltaPacket`	Une copie de la propriété DeltaPacket du composant DataSet.
`RDBMSResolver.fieldInfo`	Un nombre illimité de champs avec des propriétés identifiant les champs DataSet qui nécessitent un traitement particulier (champ-clé ou champ non-actualisable, par exemple).
`RDBMSResolver.nullValue`	L'indicateur de la valeur null d'un champ.
`RDBMSResolver.tableName`	Le nom de la table de base de données qui doit être mise à jour, dans le paquet XML.
`RDBMSResolver.updateMode`	La valeur déterminant la manière dont les champs-clés sont identifiés lorsque le paquet de mise à jour XML est généré.

Propriété	Description
RDBMSResolver.updatePacket	Une copie de la propriété updatePacket du connecteur contenant les données au format XML les plus récentes pour renvoi au composant DataSet via le connecteur (lorsque le serveur reçoit la demande de mise à jour de la part de l'application).
RDBMSResolver.updateResults	Copie de la propriété Results du connecteur, qui renvoie les éventuelles erreurs ou mises à jour (au format XML) destinées au composant DataSet.

Méthodes du composant RDBMSResolver

Méthode	Description
RDBMSResolver.addFieldInfo()	Ajoute un nouvel élément à la collection fieldInfo, afin de créer un composant Resolver de façon dynamique (à l'exécution) plutôt que d'utiliser l'inspecteur des composants dans l'environnement auteur.

Evénements du composant RDBMSResolver

Evénement	Description
RDBMSResolver.beforeApplyUpdates	Défini dans l'application ; appelé par le composant Resolver pour apporter des modifications personnalisées au paquet XML de la propriété updatePacket avant qu'il soit lié au connecteur.
RDBMSResolver.reconcileResults	Défini dans l'application ; appelé par le composant Resolver pour reconstituer les mises à jour de la propriété updatePacket envoyée au serveur et celles de la propriété updatePacket renvoyée par le serveur.
RDBMSResolver.reconcileUpdates	Défini dans l'application ; appelé par le composant Resolver pour reconstituer la mise à jour reçue par le serveur et la mise à jour en attente.

RDBMSResolver.addFieldInfo()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

donnéesResolve.addFieldInfo("*fieldName*", "*ownerName*", "*isKey*")

Paramètres

fieldName Chaîne ; fournit le nom du champ décrit par l'objet d'informations.

ownerName Chaîne ; fournit le nom de la table à laquelle le champ appartient. Peut être laissé vide ("") s'il est identique à la propriété TableName de l'occurrence resolver.

isKey Booléen ; indique s'il s'agit d'un champ clé.

Renvoie

Aucun.

Description

Méthode : ajoute un nouvel élément à la collection `fieldInfo` XML dans le paquet de mise à jour. Utilisez cette méthode si vous devez définir un composant Resolver de façon dynamique (à l'exécution) plutôt que d'utiliser l'inspecteur des composants dans l'environnement auteur.

Exemple

L'exemple suivant crée un composant Resolver et fournit le nom de la table, le nom du champ clé, et empêche la mise à jour du champ `nomTypePersonne` :

```
var monResolver:RDBMSResolver = new RDBMSResolver();
monResolver.tableName = "Clients";
// Définit le champ id en tant que champ-clé
// et empêche la mise à jour du champ nomTypePersonne.
monResolver.addFieldInfo("id", "", true);
monResolver.addFieldInfo("nomTypePersonne", "ChampJoint", false);
// Définit les liaisons de données
//...
```

RDBMSResolver.beforeApplyUpdates

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

donnéesResolve.beforeApplyUpdates(*objetEvt*)

Paramètres

objetEvt Objet événement Resolver ; décrit les personnalisations apportées au paquet XML avant l'envoi de la mise à jour à la base de données par le biais du connecteur. Cet objet événement Resolver doit contenir les propriétés suivantes :

Propriété	Description
`target`	Objet ; resolver déclenchant l'événement.
`type`	Chaîne ; nom de l'événement.
`updatePacket`	Objet XML ; l'objet XML qui sera appliqué.

Renvoie

Aucun.

Description

Propriété : propriété de type `deltaPacket` qui reçoit un `deltaPacket` à traduire en `xupdatePacket` et produit un `deltaPacket` à partir de n'importe quels résultats serveur placés dans la propriété `updateResults`. Ce gestionnaire d'événement vous permet d'apporter des modifications personnalisées au paquet XML avant l'envoi des données mises à jour au connecteur.

Les messages dans la propriété `updateResults` sont considérés comme des erreurs. Cela signifie qu'un paquet delta contenant des messages est de nouveau ajouté au `deltaPacket` pour être renvoyé lors de la prochaine transmission du `deltaPacket` au serveur. Vous devez rédiger le code qui gère les deltas contenant des messages pour que les messages soient présentés à l'utilisateur et modifiés avant d'être ajoutés au `deltaPacket` suivant.

Exemple

L'exemple suivant ajoute les données d'authentification de l'utilisateur au paquet XML :

```
on (beforeApplyUpdates) {
   // ajouter les données d'authentification de l'utilisateur
   var infoUtilisateur = new XML( ""+getUserId()+ ""+getPassword()+"" );
   updatePacket.firstChild.appendChild( infoUtilisateur );
}
```

RDBMSResolver.deltaPacket

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

`donnéesResolve``.deltaPacket`

Description

Propriété : propriété de type `deltaPacket` qui reçoit un `deltaPacket` à traduire en `updatePacket` et produit un `deltaPacket` à partir de n'importe quels résultats serveur placés dans la propriété `updateResults`.

Les messages dans la propriété `updateResults` sont considérés comme des erreurs. Cela signifie qu'un paquet delta contenant des messages est de nouveau ajouté au `deltaPacket` pour être renvoyé lors de la prochaine transmission du `deltaPacket` au serveur. Vous devez rédiger le code qui gère les deltas contenant des messages pour que les messages soient présentés à l'utilisateur et modifiés avant d'être ajoutés au `deltaPacket` suivant.

RDBMSResolver.fieldInfo

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

données Resolve.fieldInfo

Description

Propriété : propriété de type `Collection` spécifiant une collection d'un nombre illimité de champs avec des propriétés identifiant les champs DataSet qui nécessitent un traitement particulier (champs-clés ou champs non-actualisables). Chaque élément `FieldInfo` de la collection contient trois propriétés :

Propriété	Description
`FieldName`	Nom du champ de cas particulier. Le nom de ce champ doit correspondre à un nom de champ dans le DataSet.
`OwnerName`	Cette propriété facultative correspond au nom du propriétaire de ce champ s'il « n'appartient » pas à la même table que celle définie dans le paramètre `TableName` du composant. Si vous y entrez la valeur de ce paramètre ou le laissez vide, le champ est généralement inclus dans le paquet de mise à jour XML. Si vous y entrez une valeur différente, le champ est exclu du paquet de mise à jour.
`IsKey`	Valeur booléenne définie sur `true` pour tous les champs clés pour que la table soit mise à jour.

RDBMSResolver.nullValue

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

données Resolve.deltaPacket

Description

Propriété : propriété de type Chaîne utilisée pour affecter une valeur null à un champ. Vous pouvez la personnaliser pour éviter de la confondre avec une chaîne vide (`""`) ou une autre valeur valide. La chaîne par défaut est `{_NULL_}`.

RDBMSResolver.reconcileResults

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

`donnéesResolve``.reconcileResults(`*`objetEvt`*`)`

Paramètres

`objetEvt` Objet événement Resolver ; décrit l'objet événement utilisé pour comparer deux `updatePacket`s Cet objet événement Resolver doit contenir les propriétés suivantes :

Propriété	Description
`target`	Objet ; resolver déclenchant l'événement.
`type`	Chaîne ; nom de l'événement.

Renvoie

Aucun.

Description

Evénement : appelé par le composant Resolver pour comparer deux paquets une fois les résultats renvoyés par le serveur et appliqués au `deltaPacket`.

Un paquet `updateResults` peut contenir à la fois le résultat des opérations spécifiées dans le `deltaPacket` et des informations relatives aux mises à jour effectuées par d'autres clients. Lorsqu'un nouveau `updatePacket` est reçu, les résultats des opérations et les mises à jour de la base de données sont séparés en deux `updatePacket`s et placés séparément dans la propriété `deltaPacket`. L'événement `reconcileResults` est déclenché juste avant que le `deltaPacket` contenant les résultats des opérations soit envoyé à l'aide de la liaison des données.

Exemple

L'exemple suivant reconstitue deux `updatePacket`s, renvoie et efface les mises à jour en cas de succès :

```
on (reconcileResults) {
      // examiner les résultats
      if( examine( updateResults ))
        monEnsembleDeDonnées.purgeUpdates();
      else
       displayErrors( results );
}
```

RDBMSResolver.reconcileUpdates

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

resolveData.reconcileUpdates(*objetEvt*)

Paramètres

objetEvt Objet événement Resolver ; décrit les personnalisations apportées au paquet XML avant l'envoi de la mise à jour à la base de données via le connecteur. Cet objet événement Resolver doit contenir les propriétés suivantes :

Propriété	Description
target	Objet ; resolver déclenchant l'événement.
type	Chaîne ; nom de l'événement.

Renvoie

Aucun.

Description

Evénement : appelé par le composant Resolver une fois les résultats reçus du serveur, après l'application des mises à jour contenues dans un deltaPacket. Un paquet updateResults peut contenir à la fois le résultat des opérations spécifiées dans le deltaPacket et des informations relatives aux mises à jour effectuées par d'autres clients. Lorsqu'un nouveau updatePacket est reçu, les résultats d'opération et les mises à jour de base de données sont séparés en deux deltaPackets qui sont placés séparément dans la propriété deltaPacket. L'événement reconcileUpdates est déclenché juste avant que le deltaPacket contenant les résultats des opérations soit envoyé à l'aide de la liaison des données.

Exemple

L'exemple suivant reconstitue deux résultats et efface les mises à jour en cas de succès :

```
on (reconcileUpdates) {
      // examiner les résultats
      if( examine( updateResults ))
        monEnsembleDeDonnées.purgeUpdates();
      else
       displayErrors( results );
```

RDBMSResolver.tableName

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

donnéesResolve.deltaPacket

Description

Propriété : propriété de type Chaîne représentant le nom de la table de base de données qui doit être mise à jour dans le paquet XML. Elle détermine également les champs à envoyer dans l'objet `updatePacket`. A cette fin, le composant Resolver compare la valeur de cette propriété à la valeur fournie pour la propriété `fieldInfo.ownerName`. Si un champ ne comporte aucune entrée dans la propriété de collection `fieldInfo`, il est placé dans l'objet `updatePacket`. S'il comporte une entrée dans la propriété de collection `fieldInfo` et que la valeur de la propriété `ownerName` est vide ou identique à la propriété `tableName` du composant Resolver, le champ est placé dans l'objet `updatePacket`. S'il comporte une entrée dans la propriété de collection `fieldInfo` et que la valeur de la propriété `ownerName` n'est pas vide et diffère de la propriété `tableName` du composant Resolver, le champ n'est pas placé dans l'objet `updatePacket`.

RDBMSResolver.updateMode

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

donnéesResolve.deltaPacket

Description

Propriété : propriété contenant plusieurs valeurs déterminant la manière dont les champs-clés sont identifiés lorsque le paquet de mise à jour XML est généré. Cette propriété peut prendre les trois chaînes suivantes pour valeur :

Valeur	Description
`umUsingAll`	Utilise les anciennes valeurs de tous les champs modifiés pour identifier l'enregistrement à mettre à jour. Il s'agit de la valeur la plus sûre pour la mise à jour, car elle garantit que l'enregistrement n'a pas été modifié par un autre utilisateur depuis que vous l'avez récupéré. Toutefois, elle nécessite davantage de temps et génère un paquet de mise à jour plus volumineux.
`umUsingModified`	Utilise les anciennes valeurs de tous les champs modifiés pour identifier l'enregistrement à mettre à jour. Cette valeur garantit qu'aucun autre utilisateur n'a modifié les mêmes champs de l'enregistrement depuis que vous l'avez récupéré.
`umUsingKey`	Il s'agit de la valeur par défaut. Utilise les anciennes valeurs des champs clés. Elle implique un « modèle de contrôle des accès concurrents optimiste », utilisé aujourd'hui par la plupart des systèmes de base de données, et garantit que vous modifiez l'enregistrement que vous avez récupéré de la base de données.. Vos modifications remplaceront celles qu'un autre utilisateur a apportées aux mêmes données.

RDBMSResolver.updatePacket

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

`donnéesResolve``.deltaPacket`

Description

Description

Propriété : propriété de type `XML` utilisée pour établir une liaison avec une propriété de connecteur qui renvoie le paquet de mise à jour contenant les changements traduits au serveur afin de permettre la mise à jour de la source de données. Il s'agit d'un document XML contenant le paquet des changements apportés au DataSet.

RDBMSResolver.updateResults

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

`donnéesResolve``.deltaPacket`

Description

Propriété : propriété de type `deltaPacket` contenant les résultats d'une mise à jour renvoyée par le serveur via un connecteur. Utilisez cette propriété pour transmettre les erreurs et les données mises à jour depuis le serveur vers un composant DataSet, par exemple, lorsqu'un serveur affecte de nouveaux ID à un champ affecté automatiquement. Liez cette propriété à la propriété `Results` d'un connecteur afin que celui-ci puisse recevoir les résultats d'une mise à jour et retransmettre les résultats au DataSet.

Les messages dans la propriété `updateResults` sont considérés comme des erreurs. Cela signifie qu'un paquet delta contenant des messages est de nouveau ajouté au `deltaPacket` pour être renvoyé lors de la prochaine transmission du `deltaPacket` au serveur. Vous devez rédiger le code qui gère les deltas contenant des messages pour que les messages soient présentés à l'utilisateur et modifiés avant d'être ajoutés au `deltaPacket` suivant.

API du composant RPC (Remote Procedure Call)

Nom de classe ActionScript mx.data.components.RPC

L'API du composant RPC (Remote Procedure Call) est une interface (un jeu de méthodes, propriétés et événements) pouvant être implémentée par un composant Flash MX 2004 v2. L'API du composant RPC définit un moyen aisé d'envoyer des paramètres et de recevoir des résultats d'une ressource externe (service web, par exemple).

Les composants qui implémentent l'API RPC comportent les composants WebServiceConnector et XMLConnector. Ces composants agissent comme des connecteurs entre une source de données externe, telle qu'un service Web ou un fichier XML, et un composant de l'interface utilisateur de l'application.

Un composant RPC peut appeler une seule fonction externe, lui transmettre des paramètres, et recevoir des résultats. Il peut appeler la même fonction plusieurs fois. Pour appeler plusieurs fonctions, vous devez utiliser plusieurs composants.

Propriétés de la classe de composants RPC

Propriété	Description
RPC.multipleSimultaneousAllowed	Indique si plusieurs appels peuvent être effectués simultanément.
RPC.params	Spécifie les données qui seront envoyées au serveur lors de l'exécution de la prochaine opération `trigger()`.
RPC.results	Identifie les données reçues en provenance du serveur suite à l'opération `trigger()`.
RPC.suppressInvalidCalls	Indique si un appel doit être supprimé lorsque ses paramètres ne sont pas valides.

Méthodes de la classe de composants RPC

Méthode	Description
RPC.trigger()	Lance un appel de procédure à distance.

Evénements de la classe de composant RPC

Evénement	Description
RPC.result	Diffusé lorsqu'un appel de procédure à distance (RPC) s'achève avec succès.
RPC.send	Diffusé lorsque la fonction `trigger()` est en cours, après la collecte des paramètres, mais avant la validation des données et le lancement de l'appel à distance.
RPC.status	Diffusé lorsqu'un appel de procédure à distance est lancé, pour informer l'utilisateur de l'état de l'opération.

RPC.multipleSimultaneousAllowed

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.multipleSimultaneousAllowed;
```

Description

Propriété : indique si plusieurs appels peuvent être effectués simultanément. Si elle est définie sur false, la fonction `trigger()` lance un appel si un autre appel est déjà en cours. Un événement `status` est émis, avec le code `CallAlreadyInProgress`. Si la valeur est true, alors l'appel a lieu.

Lorsque plusieurs appels ont lieu simultanément, il n'est pas garanti qu'ils se termineront dans l'ordre dans lequel ils ont été déclenchés. Flash Player peut aussi limiter le nombre d'opérations réseau simultanées. Cette limite dépend de la version et de la plate-forme.

RPC.params

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.params;
```

Description

Propriété : spécifie les données qui seront envoyées au serveur lors de l'exécution de la prochaine opération `trigger()`. Chaque composant RPC définit la manière dont les données sont utilisées et quels sont les types valides.

RPC.result

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.addEventListener("result", monObjetDécoute);
```

Description

Evénement : diffusé en cas de succès d'une opération d'appel de procédure à distance.

Le paramètre du gestionnaire d'événement est un objet comportant les champs suivants :

- `type` : la chaîne "result"
- `cible` : une référence à l'objet ayant provoqué l'événement, par exemple un composant WebServiceConnector

Vous pouvez récupérer la valeur réelle du résultat à l'aide de la propriété `results`.

RPC.results

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.results;
```

Description

Propriété : identifie les données reçues en provenance du serveur suite à l'opération `trigger()`. Chaque composant RPC définit la manière dont les données sont récupérées et quels sont les types valides. Ces données apparaissent lorsque l'opération RPC a réussi, comme indiqué par l'événement `result`. Elles sont disponibles jusqu'à la purge du composant ou jusqu'à l'opération RPC suivante.

Les données renvoyées peuvent être très volumineuses. Vous pouvez les gérer de deux manières :

- Sélectionnez un clip, un scénario ou un écran approprié en tant que parent du composant RPC. Lorsque le parent ne sera plus disponible, le stockage de ce composant pourra être utilisé pour collecter divers éléments.
- Vous pouvez affecter à tout moment la valeur null à cette propriété à l'aide d'ActionScript.

RPC.send

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.addEventListener("send", monObjetDécoute);
```

Description

Evénement : diffusé pendant le traitement d'une opération `trigger()`, après la collecte des paramètres, mais avant la validation des données et le lancement de l'appel à distance. Cet événement permet d'insérer du code destiné à modifier les paramètres avant l'appel.

Le paramètre du gestionnaire d'événement est un objet comportant les champs suivants :

- `type` : la chaîne "send"
- `cible` : une référence à l'objet ayant provoqué l'événement, par exemple un composant WebServiceConnector

Vous pouvez récupérer ou modifier les valeurs réelles des paramètres à l'aide de la propriété `params`.

RPC.status

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.addEventListener("status", monObjetDécoute);
```

Description

Evénement : diffusé lorsqu'un appel de procédure à distance est lancé, pour informer l'utilisateur de l'état de l'opération.

Le paramètre du gestionnaire d'événement est un objet comportant les champs suivants :

- `type` : la chaîne "status"
- `cible` : une référence à l'objet ayant provoqué l'événement, un composant WebServiceConnector par exemple
- `code` : une chaîne donnant le nom de la condition spécifique qui s'est produite.
- `data` : un objet dont le contenu dépend du code.

Les codes et les données associées applicables à l'événement status sont les suivants :

Code	Données	Description
StatusChange	{callsInProgress:nnn}	Cet événement se produit chaque fois qu'un appel à un service web est lancé ou se termine. L'élément "nnn" correspond au nombre d'appels en cours.
CallAlreadyInProgress	aucune donnée	Cet événement se produit si (a) la fonction `trigger()` est appelée, si (b) multipleSimultaneousAllowed est false et si (c) un appel est déjà en cours. Une fois que l'événement s'est produit, l'appel tenté est considéré comme terminé ; aucun événement "result" ou "send" ne se produit.
InvalidParams	aucune donnée	Cet événement se produit si la fonction `trigger()` a déterminé que la propriété "params" ne contenait pas de données valides. Si la propriété "suppressInvalidCalls" est définie sur true, l'appel tenté est considéré comme terminé et aucun événement "result" ou "send" ne se produit.

RPC.suppressInvalidCalls

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.suppressInvalidCalls;
```

Description

Propriété : indique si un appel doit être supprimé lorsque ses paramètres ne sont pas valides. Si la valeur est `true`, alors la fonction trigger() n'effectue aucun appel si les paramètres liés ne sont pas validés. Un événement "status" est émis, avec le code `InvalidParams`. Si la valeur est false, alors l'appel a lieu et utilise les données non valides le cas échéant.

RPC.trigger()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.trigger();
```

Description

Méthode : lance un appel de procédure à distance. Chaque composant RPC définit précisément ce que cela implique. Si l'opération réussit, ses résultats apparaissent dans la propriété `results` du composant RPC.

La méthode `trigger()` effectue les opérations suivantes :

1. Si des données sont liées à la propriété `params`, la méthode exécute toutes les liaisons pour garantir que les données disponibles sont à jour. Cela provoque également la validation des données.
2. Si les données ne sont pas valides alors que le paramètre `suppressInvalidCalls` est défini sur true, l'opération est interrompue.
3. Si l'opération continue, l'événement send se produit.
4. Le véritable appel à distance est lancé via la méthode de connexion indiquée (HTTP, par exemple).

Classe Screen (Flash Professionnel uniquement)

Héritage UIObject > UIComponent > View > Loader > Screen

Nom de classe ActionScript mx.screens.Screen

La classe Screen est la classe de base des écrans que vous créez dans le panneau Contour de l'écran de Flash MX Professionnel 2004. Les écrans sont des conteneurs de haut niveau utilisés dans la création d'applications et de présentations. Pour plus d'informations sur l'utilisation des écrans, consultez « Utilisation des écrans (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

La classe Screen contient deux sous-classes principales : Slide et Form.

La classe Slide fournit le comportement d'exécution des diaporamas. La classe Slide comporte des fonctions intégrées de navigation et de séquencement, ainsi que la capacité d'ajouter facilement des transitions entre les diapositives à l'aide des comportements. Les objets Slide conservent une notion « d'état » et permettent à l'utilisateur d'atteindre la diapositive (ou l'état) suivante/précédente : lorsque la diapositive suivante apparaît, la précédente est masquée. Pour plus d'informations sur l'utilisation de la classe Slide pour contrôler les diaporamas, consultez *Classe Slide (Flash Professionnel uniquement)*, page 506.

La classe Form fournit l'environnement d'exécution des applications de formulaires. Les formulaires peuvent se superposer. Ils peuvent contenir ou être contenus dans d'autres composants. Contrairement aux diapositives, les formulaires ne disposent d'aucune capacité de séquencement ou de navigation. Pour plus d'informations sur l'utilisation de la classe Form, consultez *Classe Form (Flash Professionnel uniquement)*, page 294.

Les fonctionnalités de la classe Screen sont communes aux diapositives et aux formulaires.

Les écrans savent comment gérer leurs enfants Chaque écran comporte une propriété intégrée qui est une collection d'écrans enfant. Cette collection varie selon la disposition de la hiérarchie d'écrans dans le panneau Contour de l'écran. Les écrans peuvent posséder n'importe quel nombre d'enfants (y compris zéro), lesquels peuvent eux-mêmes avoir des enfants.

Les écrans peuvent masquer/afficher leurs enfants Un écran reste avant tout un ensemble de clips imbriqués : il peut donc contrôler la visibilité de ses enfants. Dans les applications de formulaires, tous les enfants d'un écran sont visibles en même temps par défaut ; dans les diaporamas, les écrans apparaissent généralement l'un après l'autre.

Les écrans diffusent des événements Vous pouvez par exemple déclencher la lecture d'un son ou d'une vidéo lorsqu'un écran spécifié apparaît.

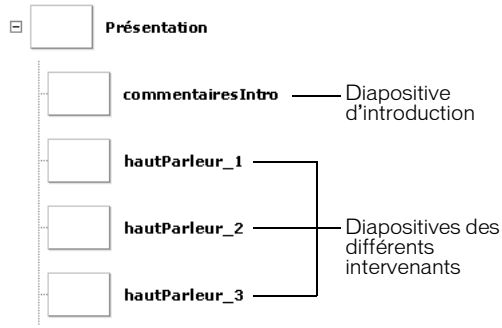
Chargement de contenu externe dans des écrans (Flash Professionnel uniquement)

La classe Screen étend la classe Loader (consultez *Composant Loader*, page 334), ce qui facilite la gestion et le chargement de fichiers SWF (et JPEG) externes. La classe Loader contient une propriété appelée `contentPath`, qui permet de spécifier l'URL d'un fichier SWF ou JPEG externe ou l'identifiant de liaison d'un clip de la bibliothèque.

Cette fonction vous permet de charger une arborescence d'écrans externe (ou tout fichier SWF externe) en tant qu'enfant de n'importe quel nœud d'écran. Cela vous permet de moduler facilement vos animations à base d'écrans et de les diviser en plusieurs fichiers SWF distincts.

Prenons l'exemple d'un diaporama réalisé par trois intervenants, qui créent chacun leur propre partie. Vous pouvez demander à chaque intervenant de créer un diaporama distinct (SWF). Il vous reste ensuite à créer un « diaporama maître » contenant trois espaces réservés, correspondant aux trois diaporamas créés par ces intervenants. Dans chaque espace réservé de diapositive, faites pointer la propriété `contentPath` vers le fichier SWF correspondant.

Vous pouvez par exemple organiser le diaporama « maître » comme illustré ci-dessous :



Structure du fichier SWF du diaporama « maître »

Supposons que les intervenants vous aient fourni trois fichiers SWF : orateur_1.swf, orateur_2.swf et orateur_3.swf. Vous pouvez facilement assembler le diaporama complet en définissant la propriété `contentPath` de chaque espace réservé de diapositive à l'aide d'ActionScript ou en définissant la propriété `contentPath` de chaque diapositive dans l'inspecteur des propriétés.

```
Orateur_1.contentPath = orateur_1.swf;
Orateur_2.contentPath = orateur_2.swf;
Orateur_3.contentPath = orateur_3.swf;
```

Vous pouvez également définir la propriété `contentPath` de chaque diapositive à l'aide de l'inspecteur des propriétés. Notez que, par défaut, lorsque vous définissez la propriété `contentPath` d'une diapositive dans l'inspecteur des propriétés (ou à l'aide du code, comme celui utilisé plus haut), le fichier SWF spécifié se charge dès que le fichier SWF du « diaporama maître » est chargé. Pour réduire le temps de chargement initial, vous pouvez définir la propriété contentPath dans un gestionnaire on(reveal) associé à chaque diapositive.

```
// Associé à la diapositive Orateur_1
on(reveal) {
  this.contentPath="orateur_1.swf";
}
```

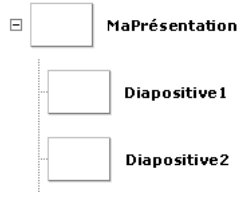
Vous pouvez également définir la propriété `autoLoad` de la diapositive (héritée de la classe Loader) sur `false`, puis appeler la méthode `load()` sur la diapositive (également héritée de la classe Loader) une fois la diapositive affichée.

```
// Associé à la diapositive Orateur_1
on(reveal) {
  this.load();
}
```

Référencement des écrans chargés avec ActionScript

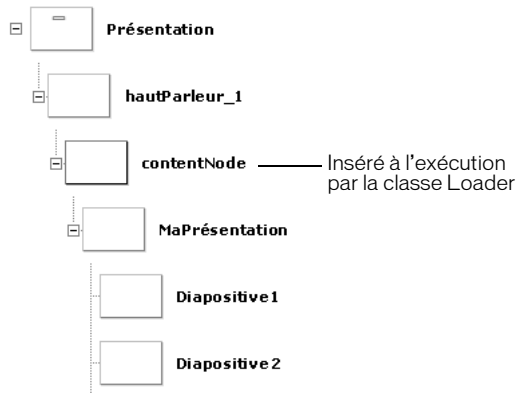
La classe Loader crée une animation interne nommée `contentNode` dans laquelle elle charge le fichier SWF ou JPEG spécifié par la propriété `contentPath`. Cette animation ajoute un nœud d'écran supplémentaire dans le diaporama chargé, entre « l'espace réservé » de la diapositive (que vous avez créé dans le diaporama « maître », ci-dessus) et la première diapositive.

Imaginons par exemple que le fichier SWF créé pour l'espace réservé de la diapositive Orateur_1 (voir l'illustration ci-dessus) présente la structure suivante, comme indiqué dans le panneau Contour de l'écran :



Structure du diaporama SWF « Orateur 1 »

A l'exécution, une fois le fichier SWF Orateur_1 chargé dans l'espace réservé de la diapositive, le diaporama complet présente la structure suivante :



Structure du diaporama « master » et « speaker » (à l'exécution)

Les propriétés et les méthodes des classes Screen, Slide et Form « ignorent » autant que possible le nœud `contentHolder`. Dans l'illustration ci-dessus, la diapositive nommée MaPrésentation (et ses sous-diapositives) fait partie de l'arborescence de diapositives contiguës qui part de la diapositive Présentation. Elle n'est pas traitée comme une sous-arborescence séparée.

Méthodes de la classe Screen

Méthode	Description
`Screen.getChildScreen()`	Renvoie l'écran enfant de cet écran à un index spécifique.

Hérite de toutes les méthodes des classes *UIObject*, *UIComponent*, SimpleButton et *Composant Loader*.

Propriétés de la classe Screen

Propriété	Description
`Screen.currentFocusedScreen`	Renvoie l'écran contenant le focus global actuel.
`Screen.indexInParent`	Renvoie l'index de l'écran (dont la numérotation commence à zéro) dans la liste des écrans enfant de son écran parent.
`Screen.numChildScreens`	Renvoie le nombre d'écrans enfant contenus dans l'écran.
`Screen.parentIsScreen`	Renvoie une valeur booléenne (`true` ou `false`) indiquant si l'objet parent de l'écran est lui-même un écran.
`Screen.rootScreen`	Renvoie l'écran racine de l'arborescence (ou sous-arborescence) contenant l'écran.

Hérite de toutes les propriétés des classes *UIObject*, *UIComponent*, View et *Composant Loader*.

Evénements de la classe Screen

Evénement	Description
`Screen.allTransitionsInDone`	Diffusé lorsque toutes les transitions « en entrée » appliquées à un écran sont terminées.
`Screen.allTransitionsOutDone`	Diffusé lorsque toutes les transitions « en sortie » appliquées à un écran sont terminées.
`Screen.mouseDown`	Diffusé lorsque l'utilisateur clique sur un objet (forme ou clip) appartenant directement à l'écran.
`Screen.mouseDownSomewhere`	Diffusé lorsque l'utilisateur clique quelque part sur la scène, mais pas nécessairement sur un objet appartenant à l'écran.
`Screen.mouseMove`	Diffusé lorsque le pointeur de la souris bouge alors qu'il se trouve sur un écran.
`Screen.mouseOut`	Diffusé lorsque le pointeur de la souris se déplace de l'intérieur vers l'extérieur de l'écran.
`Screen.mouseOver`	Diffusé lorsque le pointeur de la souris se déplace de l'extérieur vers l'intérieur de l'écran.
`Screen.mouseUp`	Diffusé lorsque l'utilisateur relâche le bouton de la souris au-dessus d'un objet (forme ou clip) appartenant directement à l'écran.
`Screen.mouseUpSomewhere`	Diffusé lorsque l'utilisateur relâche le bouton de la souris quelque part sur la scène, mais pas nécessairement sur un objet appartenant à l'écran.

Hérite de tous les événements des classes *UIObject*, *UIComponent*, View et *Composant Loader*.

Screen.allTransitionsInDone

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

```
on(allTransitionsInDone) {
  // votre code ici
}
objetDécoute = new Object();
objetDécoute.allTransitionsInDone = function(objetEvt){
  // insérez votre code ici
}
objetEcran.addEventListener("allTransitionsInDone", objetDécoute)
```

Description

Evénement : diffusé lorsque toutes les transitions « en entrée » appliquées à un écran sont terminées. L'événement `allTransitionsInDone` est diffusé par le gestionnaire de transition associé à *monEcran*.

Exemple

Dans l'exemple suivant, un bouton (`diapoSuivante_btn`) appartenant à la diapositive nommée `maDiapo` apparaît lorsque toutes les transitions « en entrée » appliquées à `maDiapo` sont terminées.

```
// Associé à maDiapo :
on(allTransitionsInDone) {
  this.diapoSuivante_btn._visible = true;
}
```

Voir aussi

`Screen.allTransitionsOutDone`

Screen.allTransitionsOutDone

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

```
on(allTransitionsOutDone) {
  // votre code ici
}
objetDécoute = new Object();
objetDécoute.allTransitionsOutDone = function(objetEvt){
  // insérez votre code ici
}
objetEcran.addEventListener("allTransitionsOutDone", objetDécoute)
```

Description

Evénement : diffusé lorsque toutes les transitions « en sortie » appliquées à l'écran sont terminées. L'événement `allTransitionsOutDone` est diffusé par le gestionnaire de transition associé à *`monEcran`*.

Voir aussi

`Screen.currentFocusedScreen`

Screen.currentFocusedScreen

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
mx.screens.Screen.currentFocusedScreen
```

Description

Propriété statique (lecture seule) : renvoie une référence à l'objet d'écran le plus éloigné dans l'arborescence contenant le focus global actuel. Le focus peut se trouver sur l'écran lui-même ou sur un clip, un objet texte ou un composant à l'intérieur de l'écran. Prend la valeur `null` par défaut s'il n'y a aucun focus actuel.

Exemple

L'exemple suivant affiche le nom de l'écran ayant actuellement le focus dans le panneau de sortie.

```
var focusActuel:mx.screens.Screen = mx.screens.Screen.currentFocusedScreen;
trace("L'écran actuel est : " + focusActuel._name);
```

Screen.getChildScreen()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

monEcran.getChildScreen(*childIndex*)

Paramètres

childIndex Nombre indiquant l'index (basé sur zéro) de l'écran enfant à renvoyer.

Renvoie

Un objet d'écran.

Description

Méthode : renvoie l'objet d'écran enfant de *monEcran* dont l'index est *childIndex*.

Exemple

L'exemple suivant affiche dans le panneau de sortie les noms de tous les écrans enfant appartenant à l'écran racine nommé `Présentation`.

```
for (var i:Number = 0; i < _root.Presentation.numChildScreens; i++) {
    var childScreen:mx.screens.Screen = _root.Presentation.getChildScreen(i);
    trace(childScreen._name);
}
```

Screen.indexInParent

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

monEcran.indexInParent

Description

Propriété (lecture seule) : contient l'index (basé sur zéro) de *monEcran* dans la liste de sous-écrans de son parent.

Exemple

L'exemple suivant affiche la position relative de l'écran monEcran dans la liste d'écrans enfant de son écran parent.

```
var numChildren:Number = monEcran._parent.numChildScreens;
var monIndex:Number = monEcran.indexInParent;
trace("Je suis la diapo enfant n° " + monIndex + " sur " + numChildren + "
  écrans.");
```

Screen.mouseDown

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

```
on(mouseDown) {
  // votre code ici
}
objetDécoute = new Object();
objetDécoute.mouseDown = function(objEvt){
  // insérez votre code ici
}
objetEcran.addEventListener("mouseDown", objetDécoute)
```

Description

Evénement : diffusé lorsque l'utilisateur clique sur un objet (par exemple, une forme ou un clip) appartenant directement à l'écran.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*objEvt*) au gestionnaire. 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 événement, consultez *Objets événement*, page 592.

Exemple

Le code suivant affiche dans le panneau de sortie le nom de l'écran ayant capturé l'événement souris.

```
on(mouseDown) {
  trace("L'événement Mouse down s'est produit sur : " + objEvt.target._name);
}
```

Screen.mouseDownSomewhere

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

```
on(mouseDown) {
  // votre code ici
}
objetDécoute = new Object();
objetDécoute.mouseDownSomewhere = function(objetEvt){
  // insérez votre code ici
}
objetEcran.addEventListener("mouseDownSomewhere", objetDécoute)
```

Description

Evénement : diffusé lorsque l'utilisateur clique sur le bouton de la souris, mais pas nécessairement sur l'écran spécifié.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*objEvt*) au gestionnaire. 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 événement, consultez *Objets événement*, page 592.

Screen.mouseMove

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

```
on(mouseDown) {
  // votre code ici
}
objetDécoute = new Object();
objetDécoute.mouseMove = function(objetEvt){
  // insérez votre code ici
}
objetEcran.addEventListener("mouseMove", objetDécoute)
```

Description

Evénement : diffusé lorsque le pointeur de la souris se déplace sur l'écran. Cet événement est uniquement envoyé lorsque la souris se trouve au-dessus du cadre de délimitation de l'écran.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*objEvt*) au gestionnaire. 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 événement, consultez *Objets événement*, page 592.

Remarque : Il peut avoir une incidence sur les performances du système et doit donc être utilisé judicieusement.

Screen.mouseOut

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

```
on(mouseOut) {
  // votre code ici
}
objetDécoute = new Object();
objetDécoute.mouseOut = function(objetEvt){
  // insérez votre code ici
}
objetEcran.addEventListener("mouseOut", objetDécoute)
```

Description

Evénement : diffusé lorsque le pointeur de la souris se déplace de l'intérieur vers l'extérieur du cadre de délimitation de l'écran.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*objEvt*) au gestionnaire. 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 événement, consultez *Objets événement*, page 592.

Remarque : Il peut avoir une incidence sur les performances du système et doit donc être utilisé judicieusement.

Screen.mouseOver

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

```
on(mouseDown) {
  // votre code ici
}
objetDécoute = new Object();
objetDécoute.mouseOver = function(objetEvt){
  // insérez votre code ici
}
objetEcran.addEventListener("mouseOver", objetDécoute)
```

Description

Evénement : diffusé lorsque le pointeur de la souris se déplace de l'extérieur vers l'intérieur du cadre de délimitation de l'écran.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*objEvt*) au gestionnaire. 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 événement, consultez *Objets événement*, page 592.

Remarque : Il peut avoir une incidence sur les performances du système et doit donc être utilisé judicieusement.

Screen.mouseUp

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

```
on(mouseUp) {
  // votre code ici
}
objetDécoute = new Object();
objetDécoute.mouseUP = function(objetEvt){
  // insérez votre code ici
}
objetEcran.addEventListener("mouseUp", objetDécoute)
```

Description

Evénement : diffusé lorsque le bouton de la souris est relâché au-dessus de l'écran.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*objEvt*) au gestionnaire. 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 événement, consultez *Objets événement*, page 592.

Screen.mouseUpSomewhere

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

```
on(mouseUpSomewhere) {
  // votre code ici
}
objetDécoute = new Object();
objetDécoute.mouseUpSomewhere = function(objetEvt){
  // insérez votre code ici
}
objetEcran.addEventListener("mouseUpSomewhere", objetDécoute)
```

Description

Evénement : diffusé lorsque l'utilisateur clique sur le bouton de la souris, mais pas nécessairement au-dessus de l'écran spécifié.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*objEvt*) au gestionnaire. 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 événement, consultez *Objets événement*, page 592.

Screen.numChildScreens

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

```
monEcran.numChildScreens
```

Description

Propriété (lecture seule) : renvoie le nombre d'écrans enfant contenus dans *monEcran*.

Exemple

L'exemple suivant affiche les noms de tous les écrans enfant appartenant à `monEcran`.

```
var nombreEnfants:Number = monEcran.numChildScreens;
for(i=0; i<combienDEnfants; i++) {
  var écranEnfant = monEcran.getChildScreen(i);
  trace(écranEnfant._name);
}
```

Voir aussi

`Screen.getChildScreen()`

Screen.parentIsScreen

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`monEcran``.parentIsScreen`

Description

Propriété (lecture seule) : renvoie une valeur booléenne (`true` ou `false`) indiquant si l'objet parent de l'écran spécifié est également un écran (`true`) ou non (`false`). Si la valeur est false, *`monEcran`* se trouve en haut de sa hiérarchie d'écrans.

Exemple

Le code suivant détermine si l'objet parent de l'écran `monEcran` est également un écran. Si c'est le cas, `monEcran` est supposé être la diapositive racine, ou maître, du diaporama et ne pas posséder de frères. Si `monEcran.parentIsScreen` renvoie `true`, le nombre d'écrans frères de `monEcran` apparaît dans le panneau de sortie.

```
if (monEcran.parentIsScreen) {
  trace("J'ai "+monEcran._parent.numChildScreens+" écran(s) frère(s)");
} else {
  trace("Je suis l'écran racine et je n'ai pas de frère");
}
```

Screen.rootScreen

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`monEcran``.rootScreen`

Description

Propriété (lecture seule) : renvoie l'écran situé en haut de la hiérarchie d'écrans et contenant *monEcran*.

Exemple

L'exemple suivant affiche le nom de l'écran racine contenant l'écran monEcran.

```
var maRacine:mx.screens.Screen = monEcran.rootScreen;
```

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 27 ou *Classe FocusManager*, page 287.

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 avec la souris le contenu à l'intérieur du panneau ; un curseur en forme de main apparaît sur le contenu. Contrairement à la plupart des autres composants, les événements sont diffusés lorsque le bouton de la souris est enfoncé et continuent à diffuser jusqu'à ce que l'utilisateur relâche le bouton de la souris. Si le contenu d'un panneau défilant comporte des arrêts de tabulation valides, vous devez définir `scrollDrag` sur `false` sinon chaque interaction de la souris avec les contenus invoquera le déplacement par défilement.

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 programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez la méthode `setSize()` (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 de styles avec le composant ScrollPane

Le composant ScrollPane ne gère pas les styles, contrairement aux barres de défilement qu'il utilise.

Utilisation des enveloppes avec le composant ScrollPane

Le composant ScrollPane ne dispose d'aucune enveloppe contrairement aux barres de défilement qu'il utilise.

Classe ScrollPane

Héritage UIObject > UIComponent > View > ScrollView > ScrollPane

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 d'informations 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 des classes *UIObject* et *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.

Méthode	Description
`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 des classes *UIObject* et *UIComponent*.

Evénements de la classe ScrollPane

Méthode	Description
`ScrollPane.complete`	Diffusé lorsque le contenu du panneau défilant est chargé.
`ScrollPane.progress`	Diffusé lorsque le contenu du panneau défilant est en cours de chargement.
`ScrollPane.scroll`	Diffusé lorsque la barre de défilement est enfoncée.

Hérite de tous les événements des classes *UIObject* et *UIComponent*.

ScrollPane.complete

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(complete){
  ...
}
```

Usage 2 :

```
objetDécoute = new Object();
ObjetDécoute.complete = function(objetEvt){
  ...
}
monPanneauDéfilant.addEventListener("complete", objetDécoute)
```

Description

Evénement : diffusé 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 (également appelée *gestionnaire*) sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

Exemple

L'exemple suivant crée un objet d'écoute avec un gestionnaire d'événement `complete` pour l'occurrence de `ScrollPane` :

```
form.complete = function(objEvt){
  // insérez le code afin de gérer l'événement
}
scrollPane.addEventListener("complete",form);
```

ScrollPane.content

Disponibilité

Flash Player 6 version 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;
```

Voir aussi

ScrollPane.contentPath

ScrollPane.contentPath

Disponibilité

Flash Player 6 version 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";
```

Voir aussi

`ScrollPane.content`

ScrollPane.getBytesLoaded()

Disponibilité

Flash Player 6 version 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(objEvt){
  // objEvt.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 version 79.

Edition

Flash MX 2004.

Usage

`OccurrenceDePanneauDéfilant``.getBytesTotal()`

Paramètres

Aucun.

Renvoie

Un nombre.

Description

Méthode : renvoie le nombre total d'octets à charger dans l'occurrence monPanneauDéfilant.

Voir aussi

`ScrollPane.getBytesLoaded()`

ScrollPane.hLineScrollSize

Disponibilité

Flash Player 6 version 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 version 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 version 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 version 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 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(progress){
   ...
}
```

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.progress = function(objetEvt){
   ...
}
OccurrenceDePanneauDéfilant.addEventListener("progress", objetDécoute)
```

Description

Evénement : diffusé à l'ensemble des écouteurs enregistrés pendant le chargement du contenu. L'événement progress n'est pas toujours diffusé. L'événement `complete` peut être diffusé sans qu'aucun événement `progress` ne soit distribué. Ceci peut particulièrement se produire si le contenu chargé est un fichier local. 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 (également appelée *gestionnaire*) sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

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(objEvt){
  // objEvt.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 version 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 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(scroll){
  ...
}
```

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.scroll = function(objetEvt){
  ...
}
OccurrenceDePanneauDéfilant.addEventListener("scroll", objetDécoute)
```

Objet événement

Outre les propriétés standard de l'objet événement, une propriété `type` est définie pour l'événement `scroll` : elle prend la valeur `"scroll"`. Il existe également une propriété `direction` pouvant avoir les valeurs `"vertical"` et `"horizontal"`.

Description

Evénement : diffusé 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 (également appelée *gestionnaire*) sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

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(objEvt){
  trace("Panneau défilant parcouru");
}
occurrencePd.addEventListener("scroll", form);
```

Voir aussi

`UIEventDispatcher.addEventListener()`

ScrollPane.scrollDrag

Disponibilité

Flash Player 6 version 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 l'occurrence de composant 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 version 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 version 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 version 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 version 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 Slide (Flash Professionnel uniquement)

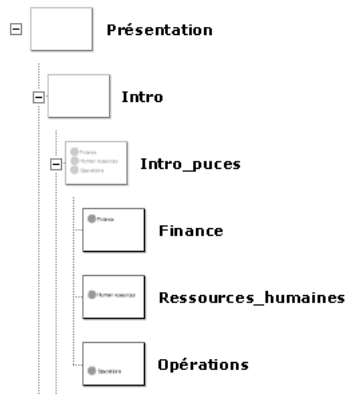
Héritage UIObject > UIComponent > View > Loader > Screen > Slide

Nom de classe ActionScript mx.screens.Slide

La classe Slide correspond à un nœud dans un diaporama hiérarchique. Dans Flash MX Professionnel 2004, le panneau Contour de l'écran vous permet de créer des diaporamas. Pour plus d'informations sur l'utilisation des écrans, consultez « Utilisation des écrans (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

La classe Slide étend la classe Screen (consultez *Classe Screen (Flash Professionnel uniquement)*, page 477). En plus de fournir des fonctions intégrées de navigation et de séquencement entre les diapositives, elle permet d'ajouter facilement des transitions entre les diapositives à l'aide des comportements. Les objets Slide conservent une notion « d'état » qui permet à l'utilisateur d'atteindre la diapositive suivante ou précédente d'un diaporama. Lorsque la diapositive suivante apparaît, la diapositive précédente est masquée.

Notez que vous pouvez uniquement accéder aux diapositives (ou vous « arrêter » sur les diapositives) ne contenant pas de diapositives enfant (ou « feuille »). L'illustration suivante affiche le contenu du panneau Contour de l'écran pour un exemple de diaporama.



Lorsque le diaporama commence, il « s'arrête » par défaut sur la diapositive appelée Finance, qui correspond à la première diapositive du diaporama ne contenant pas de diapositive enfant.

Notez que les diapositives enfant « héritent » de l'apparence visuelle (graphiques et autre contenu) de leurs diapositives parent. Par exemple, dans l'illustration ci-dessus, en plus du contenu de la diapositive Finance, l'utilisateur voit le contenu des diapositives Intro et Présentation.

Remarque : La classe Slide hérite de la classe Loader (consultez *Classe Loader*, page 336), qui permet de charger facilement des fichiers SWF (ou JPEG) dans une diapositive donnée. Cela permet de modulariser les diaporamas et de réduire le délai initial de téléchargement. Pour plus d'informations, consultez *Chargement de contenu externe dans des écrans (Flash Professionnel uniquement)*, page 478.

Utilisation de la classe Slide (Flash Professionnel uniquement)

Utilisez les méthodes et propriétés de la classe Slide pour contrôler les diaporamas créés à l'aide du panneau Contour de l'écran (Fenêtre > Ecrans), pour obtenir des informations sur un diaporama (par exemple, déterminer le nombre de diapositives enfant contenues dans la diapositive parent) ou pour naviguer entre les diapositives d'un diaporama (par exemple, pour créer les boutons « Diapositive suivante » et « Diapositive précédente »).

Vous pouvez également utiliser l'un des comportements intégrés disponibles dans le panneau Comportements (Fenêtre > Panneaux de développement > Comportements) pour contrôler les diaporamas. Pour plus d'informations sur l'utilisation des comportements avec les diapositives, consultez « Ajout de commandes sur les écrans à l'aide des comportements (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Paramètres du composant Slide

Vous pouvez définir les paramètres de programmation suivants pour chaque diapositive dans l'inspecteur des propriétés ou le panneau des composants :

autoKeyNav détermine si la diapositive réagit aux interactions clavier par défaut et de quelle manière. Pour plus d'informations, consultez `Slide.autoKeyNav`.

autoload indique si le contenu spécifié par le paramètre `contentPath` doit être chargé automatiquement (true) ou attendre que la méthode `Loader.load()` soit appelée (false) pour se charger. La valeur par défaut est true.

contentPath spécifie le contenu de la diapositive. Il peut s'agir de l'identificateur de liaison d'un clip ou d'une URL absolue ou relative d'un fichier SWF ou JPG à charger dans la diapositive. Par défaut, le contenu chargé est coupé pour être adapté à la diapositive.

overlayChildren spécifie si les diapositives enfant de la diapositive restent visibles (true) ou disparaissent (false) lorsque vous passez d'une diapositive enfant à la suivante.

playHidden spécifie si la lecture de la diapositive continue lorsqu'elle est masquée (true) ou si elle s'arrête (false).

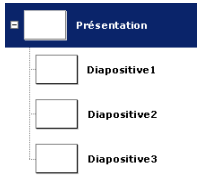
Utilisation de la classe Slide pour créer un diaporama

Utilisez les méthodes et les propriétés de la classe Slide pour contrôler les diaporamas que vous créez dans le panneau Contour de l'écran (Fenêtre > Ecrans) de l'environnement auteur de Flash. Notez que le panneau Comportements (Fenêtre > Panneaux de développement > Comportements) contient plusieurs comportements permettant de créer et configurer la navigation entre des diapositives. Dans cet exemple, vous écrivez votre propre code ActionScript pour créer les boutons Suivant et Précédent d'un diaporama.

Pour créer un diaporama avec des commandes de navigation :

1. Dans Flash, choisissez Fichier > Nouveau.
2. Cliquez sur l'onglet Général et sélectionnez Diaporama Flash dans la section Type.
3. Dans le panneau Contour de l'écran, cliquez deux fois sur le bouton Insérer un écran (+) pour créer deux nouvelles diapositives sous la diapositive Présentation.

Le panneau Contour de l'écran doit avoir l'aspect suivant :



4 Sélectionnez la Diapositive 1 dans le panneau Contour de l'écran et ajoutez un champ de texte contenant « Ceci est la diapositive n°1 » à l'aide de l'outil Texte.

5 Répétez cette opération pour les diapositives 2 et 3, en créant deux champs de texte contenant « Ceci est la diapositive n°2 » et « Ceci est la diapositive n°3 » respectivement.

6 Sélectionnez le diaporama et ouvrez le panneau Composants (Fenêtre > Panneaux de développement > Composants).

7 Faites glisser un composant Button du panneau Composants vers la partie inférieure de la scène.

8 Dans l'inspecteur des propriétés (Fenêtre > Propriétés), entrez « Diapositive suivante » pour la propriété Label du composant Button.

9 Ouvrez le panneau Actions, s'il n'est pas déjà ouvert, en sélectionnant Fenêtre > Panneaux de développement > Actions.

10 Tapez le code suivant dans le panneau Actions :

```
on(click) {
  _parent.currentSlide.gotoNextSlide();
}
```

11 Testez le fichier SWF (Contrôle > Tester l'animation) et cliquez sur le bouton Diapositive suivante pour atteindre la diapositive suivante.

Méthodes de la classe Slide

Propriété	Description
`Slide.getChildSlide()`	Renvoie la diapositive enfant de cette diapositive à un index donné.
`Slide.gotoFirstSlide()`	Permet d'atteindre le premier nœud feuille dans la hiérarchie de sous-diapositives de la diapositive.
`Slide.gotoLastSlide()`	Permet d'atteindre le dernier nœud feuille dans la hiérarchie de sous-diapositives de la diapositive.
`Slide.gotoNextSlide()`	Permet d'atteindre la diapositive suivante.
`Slide.gotoPreviousSlide()`	Permet d'atteindre la diapositive suivante.
`Slide.gotoSlide()`	Permet d'atteindre une diapositive spécifique.

Hérite de toutes les méthodes des classes *UIObject*, *UIComponent*, View, *Composant Loader* et de la classe *Classe Screen (Flash Professionnel uniquement)*.

Propriétés de la classe Slide

Propriété	Description
`Slide.autoKeyNav`	Détermine si la diapositive réagit ou non aux interactions clavier par défaut pour naviguer vers la diapositive suivante ou précédente.
`Slide.currentSlide`	Renvoie le premier enfant de la diapositive contenant la diapositive active.
`Slide.currentSlide`	Renvoie la diapositive active.
`Slide.currentFocusedSlide`	Renvoie la diapositive la plus éloignée dans l'arborescence et contenant le focus global actuel.
`Slide.defaultKeydownHandler`	Gestionnaire de rappel qui prend la priorité sur les interactions clavier par défaut utilisées pour la navigation entre les diapositives (flèches gauche et droite).
`Slide.firstSlide`	Renvoie la première diapositive enfant de la diapositive n'ayant pas d'enfant.
`Slide.getChildSlide()`	Renvoie la diapositive enfant à l'index spécifié.
`Slide.indexInParentSlide`	Renvoie l'index de la diapositive (la numérotation de l'index commence à zéro) dans la liste de sous-diapositives de son parent.
`Slide.lastSlide`	Renvoie la dernière diapositive enfant de la diapositive n'ayant pas d'enfant.
`Slide.nextSlide`	Renvoie la diapositive du nœud feuille suivant.
`Slide.numChildSlides`	Renvoie le nombre de diapositives enfant de la diapositive.
`Slide.overlayChildren`	Détermine si les diapositives enfant de la diapositive sont visibles lorsque le contrôle passe d'une diapositive enfant à la suivante.
`Slide.parentIsSlide`	Renvoie une valeur booléenne indiquant si l'objet parent de la diapositive est également une diapositive (`true`) ou non (`false`).
`Slide.playHidden`	Détermine si la lecture de la diapositive continue lorsqu'elle est masquée.
`Slide.previousSlide`	Renvoie la diapositive du nœud feuille précédent.
`Slide.revealChild`	Renvoie la racine de l'arborescence de diapositives contenant la diapositive.

Hérite de toutes les propriétés des composants *UIObject*, *UIComponent*, View, *Composant Loader* et de la *Classe Screen (Flash Professionnel uniquement)*.

Evénements de la classe Slide

Evénement	Description
`Slide.hideChild`	Diffusé lorsque tous les enfants d'une diapositive passent de l'état visible à l'état invisible.
`Slide.revealChild`	Diffusé lorsque tous les enfants d'une diapositive passent de l'état invisible à l'état visible.

Hérite de tous les événements des classes *UIObject*, *UIComponent*, View, *Composant Loader* et *Classe Screen (Flash Professionnel uniquement)*.

Slide.autoKeyNav

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

maDiapo.autoKeyNav

Description

Propriété : détermine si la diapositive réagit ou non aux interactions clavier par défaut pour naviguer vers la diapositive suivante ou précédente lorsque *maDiapo* a le focus. Cette propriété peut prendre l'une des valeurs de chaînes suivantes : `"true"`, `"false"` ou `"inherit"`. Vous pouvez également modifier le comportement de traitement clavier par défaut à l'aide de la propriété `Slide.defaultKeydownHandler`.

Vous pouvez également définir cette propriété à l'aide de l'inspecteur des propriétés.

Lorsque la propriété est définie sur `"true"`, une pression sur la flèche droite (`Key.RIGHT`) ou sur la barre d'espace (`Key.SPACE`) alors que *maDiapo* a le focus permet d'accéder à la diapositive suivante ; une pression sur la flèche gauche (`Key.Left`) permet de revenir à la diapositive précédente.

Lorsque la propriété est définie sur `"false"`, aucune interaction clavier par défaut ne se produit lorsque *maDiapo* a le focus.

Lorsque la propriété est définie sur `"inherit"`, *maDiapo* examine la propriété `autoKeyNav` de sa diapositive parent. Si cette propriété est également définie sur `"inherit"` pour le parent de *maDiapo*, le parent du parent de *maDiapo* est examiné à son tour et ainsi de suite, jusqu'à ce qu'une diapositive dont la propriété `autoKeyNav` est définie sur `"true"` ou `"false"` soit trouvée.

Si *maDiapo* n'a pas de diapositive parent (si `(maDiapo.parentIsSlide == false)` est `true`), elle se comporte comme si `autoKeyNav` avait été défini sur `true`.

Exemple

Cet exemple désactive la navigation clavier automatique pour la diapositive nommée `diapoConnexion`.

```
_root.Présentation.diapoConnexion.autoKeyNav = "false";
```

Voir aussi

`Slide.defaultKeydownHandler`

Slide.currentSlide

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`maDiapo``.currentSlide`

Description

Propriété (lecture seule) : renvoie la diapositive active. Il s'agit toujours d'une diapositive « feuille », c'est-à-dire une diapositive ne contenant aucune diapositive enfant.

Exemple

Le code suivant, associé à un bouton de la diapositive Présentation racine, permet de passer à la diapositive suivante du diaporama à chaque fois que l'utilisateur clique sur le bouton.

```
// Associé à une occurrence de bouton contenue dans la diapositive
  Présentation :
on(press) {
  _parent.currentSlide.gotoNextSlide();
}
```

Voir aussi

`Slide.gotoNextSlide()`

Slide.currentChildSlide

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`maDiapo``.currentChildSlide`

Description

Propriété (lecture seule) : renvoie le premier enfant de *`maDiapo`* contenant la diapositive active ; renvoie `null` si aucune diapositive enfant contenue dans *`maDiapo`* n'a le focus actuel.

Exemple

Considérons par exemple le contour de l'écran suivant :

```
Présentation
  Diapo_1
    Puce1_1
      SousPuce1_1_1
    Puce1_2
      SousPuce1_2_1
  Diapo_2
```

Si `SousPuce1_1_1` est la diapositive actuelle, toutes les affirmations suivantes sont vraies :

```
Présentation.currentChildSlide == Diapo_1;
Diapo_1.currentChildSlide == Puce_1_1;
SousPuce_1_1_1.currentChildSlide == null;
Diapo_2.currentChildSlide == null;
```

Voir aussi

`Slide.currentSlide`

Slide.currentFocusedSlide

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

```
mx.screens.Slide.currentFocusedSlide
```

Description

Propriété (lecture seule) : renvoie la diapositive la plus éloignée dans l'arborescence et contenant le focus global actuel. Le focus actuel peut se trouver dans la diapositive elle-même, dans un clip, dans un objet texte ou dans un composant de la diapositive ; la propriété renvoie `null` s'il n'y a aucun focus actuel.

Exemple

```
var diapoFocus = mx.screens.Slide.currentFocusedSlide;
```

Slide.defaultKeydownHandler

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

```
maDiapo.defaultKeyDownHandler = function (objEvt) {
  // votre code ici
}
```

Paramètres

objEvt Objet événement avec les propriétés suivantes :

- type Chaîne indiquant le type d'événement. Les valeurs possibles sont "keyUp" et "keyDown".
- ascii Entier représentant la valeur ASCII de la dernière touche enfoncée ; correspond à la valeur renvoyée par Key.getAscii().
- code Entier représentant le code de touche de la dernière touche enfoncée ; correspond à la valeur renvoyée par Key.getCode().
- shiftKey Valeur booléenne (true ou false) indiquant si la touche Maj est enfoncée (true) ou non (false).
- ctrlKey Valeur booléenne (true ou false) indiquant si la touche Ctrl est enfoncée (true) ou non (false).

Renvoie

Rien.

Description

Gestionnaire de rappel : permet de remplacer la touche de navigation par défaut du clavier par un gestionnaire de clavier personnalisé que vous créez. Par exemple, plutôt que d'utiliser les flèches gauche et droite pour naviguer respectivement vers les diapositives précédente et suivante d'un diaporama, vous pouvez décider d'utiliser les flèches Haut et Bas. Pour plus d'informations concernant le comportement de traitement des interactions clavier par défaut, consultez Slide.autoKeyNav.

Le traitement automatique des interactions clavier est activé lorsque la propriété Slide.autoKeyNav de la diapositive actuelle est définie sur "true". C'est également le cas si cette propriété est définie sur "inherit" et que l'ancêtre le plus immédiat de la diapositive actuelle n'ayant pas la propriété "inherit" est la diapositive racine du diaporama ou possède une valeur autoKeyNav définie sur "true".

Exemple

Dans cet exemple, le traitement des interactions clavier par défaut est modifié pour les diapositives enfant de la diapositive à laquelle le gestionnaire on(load) est associé. Le gestionnaire utilise les flèches de navigation Haut/Bas au lieu des flèches Gauche/Droite.

```
on (load) {
  this.defaultKeyDownHandler = function(objEvt:Object) {
    switch (objEvt.code) {
    case Key.DOWN :
      this.currentSlide.gotoNextSlide();
      break
    case Key.UP :
      this.currentSlide.gotoPreviousSlide();
      break
    default :
      break
    }
  };
}
```

Voir aussi

`Slide.autoKeyNav`

Slide.firstSlide

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

maDiapo.firstSlide

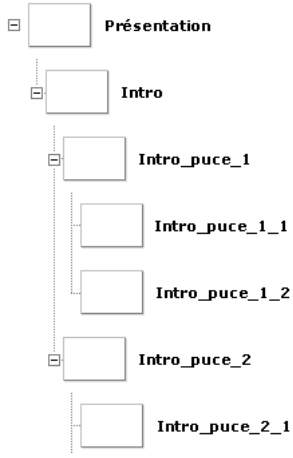
Description

Propriété (lecture seule) : renvoie la première diapositive enfant de *maDiapo* n'ayant pas d'enfant.

Exemple

Par exemple, dans la hiérarchie de diapositives illustrée ci-dessous, toutes les affirmations suivantes sont vraies :

```
Présentation.Intro.firstSlide == Intro_puce_1_1;
Présentation.Intro_puce_1.firstSlide == Intro_puce_1-1;
```



Slide.getChildSlide()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

maDiapo.getChildSlide(*indexEnfant*)

Paramètres

indexEnfant Index de la diapositive enfant à renvoyer ; la numérotation de l'index commence à zéro.

Renvoie

Un objet diapositive.

Description

Méthode : renvoie la diapositive enfant de *maDiapo* dont l'index correspond à *indexEnfant*. Cette méthode permet par exemple d'itérer sur un jeu de diapositives enfant dont les index sont connus, comme dans l'exemple suivant.

Exemple

Cet exemple affiche dans le panneau de sortie les noms de toutes les diapositives enfant de la diapositive Présentation racine.

```
var nombreDiapos = _root.Présentation.numChildSlides;
for(var indexDiapo=0; indexDiapo < nombreDiapos; indexDiapo++) {
  var diapoEnfant = _root.Présentation.getChildSlide(indexDiapo);
  trace(diapoEnfant._name);
}
```

Voir aussi

Slide.numChildSlides

Slide.gotoSlide()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

maDiapo.gotoSlide(*nouvelleDiapo*)

Paramètres

nouvelleDiapo Diapositive à atteindre.

Renvoie

Une valeur booléenne (`true` ou `false`) indiquant si la navigation a réussi (`true`) ou non (`false`).

Description

Méthode : permet d'atteindre la diapositive spécifiée par *`nouvelleDiapo`*. Pour que la navigation réussisse, les conditions suivantes doivent être remplies :

- La diapositive actuelle doit être un enfant de *`maDiapo`*.
- La diapositive spécifiée par *`nouvelleDiapo`* et la diapositive actuelle doivent avoir un ancêtre de diapositive commun, ce qui signifie que la diapositive actuelle et *`nouvelleDiapo`* doivent se trouver dans la même sous-arborescence de diapositives.

Si aucune de ces conditions n'est remplie, la navigation échoue et la méthode renvoie `false` ; sinon, la méthode atteint la diapositive spécifiée et renvoie `true`.

Considérons par exemple la hiérarchie de diapositives suivante :

```
Présentation
  Diapo1
    Diapo1_1
    Diapo1_2
 Diapo2
    Diapo2_1
    Diapo2_2
```

Si la diapositive actuelle est `Diapo1_2`, alors la méthode `gotoSlide()` suivante échouera, puisque la diapositive actuelle n'est pas un descendant de `Diapo2` :

```
Diapo2.gotoSlide(Diapo2_1);
```

Considérons également la hiérarchie de diapositives suivante, dans laquelle un objet de formulaire est l'écran parent de deux arborescences de diapositives distinctes.

```
Formulaire_1
  Diapo1
    Diapo1_1
    Diapo1_2
  Diapo2
    Diapo2_1
    Diapo2_2
```

Si la diapositive actuelle est `Diapo1_2`, alors la méthode suivante échouera également car `Diapo1` et `Diapo2` se situent dans des sous-arborescences de diapositives différentes.

```
Diapo1_2.gotoSlide(Diapo2_2);
```

Exemple

Le code suivant, associé à un composant Button, utilise la propriété `Slide.currentSlide` et la méthode `gotoSlide()` pour atteindre la diapositive suivante dans le diaporama.

```
on(click) {
  _parent.gotoSlide(_parent.currentSlide.nextSlide);
}
```

Notez que cela équivaut au code suivant, utilisant la méthode `Slide.gotoNextSlide()` :

```
on(click) {
  _parent.currentSlide.gotoNextSlide();
}
```

Voir aussi

`Slide.currentSlide`, `Slide.gotoNextSlide()`

Slide.gotoFirstSlide()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`maDiapo``.gotoFirstSlide()`

Renvoie

Rien.

Description

Méthode : permet d'atteindre la première diapositive feuille dans l'arborescence de diapositives enfant sous *`maDiapo`*. Cette méthode est ignorée lorsqu'elle est appelée depuis le gestionnaire d'événement `on(hide)` ou `on(reveal)` d'une diapositive, si l'événement résulte d'une opération de navigation dans les diapositives.

Pour atteindre la première diapositive du diaporama, appelez *`maDiapo`*`.rootSlide.gotoFirstSlide()`. Pour plus d'informations sur `rootSlide`, consultez `Slide.revealChild`.

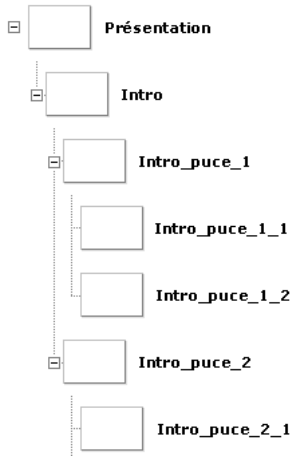
Exemple

Dans la hiérarchie de diapositives illustrée ci-dessous, les appels de méthode suivants permettent tous d'atteindre la diapositive nommée `Intro_puce_1_1`.

```
Présentation.gotoFirstSlide();
Présentation.Intro.gotoFirstSlide();
Présentation.Intro.Intro_puce_1.gotoFirstSlide();
```

Cette méthode permet d'atteindre la diapositive nommée `Intro_puce_2_1`.

```
Présentation.Intro.Intro_puce_2.gotoFirstSlide();
```



Voir aussi

`Slide.firstSlide`, `Slide.revealChild`

Slide.gotoLastSlide()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`maDiapo``.gotoLastSlide()`

Renvoie

Rien.

Description

Méthode : permet d'atteindre la dernière diapositive feuille dans l'arborescence de diapositives enfant sous *`maDiapo`*. Cette méthode est ignorée lorsqu'elle est appelée depuis le gestionnaire d'événement `on(hide)` ou `on(reveal)` d'une diapositive, si l'événement résulte d'une opération de navigation dans les diapositives.

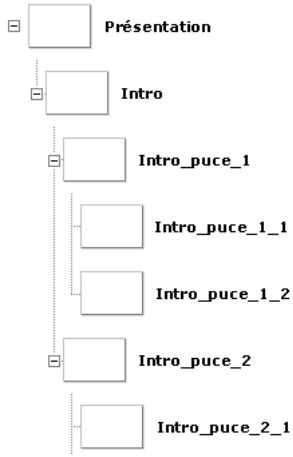
Exemple

Dans la hiérarchie de diapositives illustrée ci-dessous, les appels de méthode suivants permettent d'atteindre la diapositive nommée `Intro_puce_1_2`.

```
Présentation.Intro.gotoLastSlide();
Présentation.Intro.Intro_puce_1.gotoLastSlide();
```

Ces appels de méthode permettent d'atteindre la diapositive nommée `Intro_puce_2_1`.

```
Présentation.gotoLastSlide();
Présentation.Intro.gotoLastSlide();
```



Voir aussi

`Slide.gotoSlide()`, `Slide.lastSlide`

Slide.gotoNextSlide()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

maDiapo`.gotoNextSlide()`

Renvoie

Valeur booléenne (`true` ou `false`) ou `null` : renvoie `true` si la méthode a permis d'atteindre la diapositive suivante ; renvoie `false` si la dernière diapositive du diaporama avait déjà été atteinte lors de l'appel de la méthode (si `currentSlide.nextSlide` est `null`) ; renvoie `null` si la méthode est appelée sur une diapositive qui ne contient pas la diapositive actuelle.

Description

Méthode : permet d'atteindre la diapositive suivante du diaporama. Lorsque le contrôle passe d'une diapositive à la suivante, la diapositive sortante est masquée et la diapositive entrante est affichée. Si les diapositives se trouvent dans des sous-arborescences de diapositives différentes, toutes les diapositives ancêtre, depuis la diapositive sortante jusqu'à l'ancêtre commun des diapositives entrante et sortante, sont masquées et reçoivent un événement `hide`. Tout de suite après, toutes les diapositives ancêtre de la diapositive entrante, jusqu'à l'ancêtre commun des diapositives entrante et sortante, sont affichées et reçoivent un événement `reveal`.

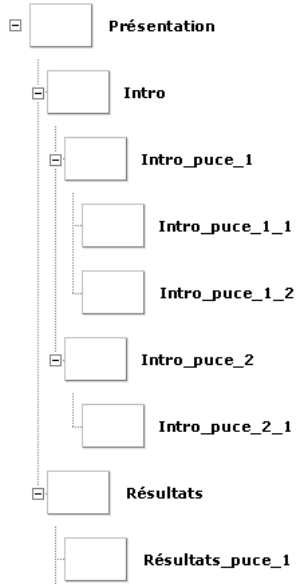
La méthode `gotoNextSlide()` est généralement appelée sur le nœud feuille représentant la diapositive actuelle. Si elle est appelée sur un nœud non feuille, par exemple `unNœud`, la méthode `unNœud.gotoNextSlide()` permet d'atteindre le premier nœud feuille de la diapositive, ou « section », suivante. Pour plus d'informations, consultez l'exemple ci-dessous.

Cette méthode n'a aucun effet lorsqu'elle est appelée sur une diapositive qui ne contient pas la diapositive actuelle (voir l'exemple ci-dessous).

Cette méthode n'a pas plus d'effet lorsqu'elle est appelée depuis un gestionnaire d'événement `on(hide)` ou `on(reveal)` associé à une diapositive, si ce gestionnaire a été appelé suite à une opération de navigation dans les diapositives.

Exemple

Supposons que, dans la hiérarchie de diapositives suivante, la diapositive nommée `Intro_puce_1_1` soit la diapositive actuellement affichée (c'est-à-dire que `_root.Présentation.currentSlide._name == Intro_puce_1_1`).



Dans ce cas, l'appel de `Intro_puce_1_1.gotoNextSlide()` permet d'atteindre `Intro_puce_1_2`, qui est un frère de `Intro_puce_1_1`.

Cependant, l'appel de `Intro_puce_1.gotoNextSlide()` permet d'atteindre `Intro_puce_2_1`, la première diapositive feuille contenue dans `Intro_puce_2`, qui est le frère suivant de `Intro_puce_1`. De même, l'appel de `Intro.gotoNextSlide()` permet d'atteindre `Résultats_puce_1`, la première diapositive feuille contenue dans la diapositive `Résultats`.

Enfin, en supposant toujours que la diapositive actuelle est `Intro_puce_1_1`, l'appel de `Résultats.gotoNextSlide()` n'a aucun effet, puisque `Résultats` ne contient pas la diapositive actuelle (ce qui signifie que `Résultats.currentSlide` est `null`).

Voir aussi

`Slide.currentSlide`, `Slide.gotoPreviousSlide()`, `Slide.nextSlide`

Slide.gotoPreviousSlide()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

maDiapo.gotoPreviousSlide()

Renvoie

Valeur booléenne (true ou false) ou null : renvoie true si la méthode a permis d'atteindre la diapositive précédente ; renvoie false si le diaporama se trouve sur la première diapositive lors de l'appel de la méthode (si currentSlide.nextSlide est null) ; renvoie null si la méthode est appelée sur une diapositive qui ne contient pas la diapositive actuelle.

Description

Méthode : permet d'atteindre la diapositive précédente du diaporama. Lorsque le contrôle passe d'une diapositive à la précédente, la diapositive sortante est masquée et la diapositive entrante est affichée. Si les diapositives se trouvent dans des sous-arborescences de diapositives différentes, toutes les diapositives ancêtre, depuis la diapositive sortante jusqu'à l'ancêtre commun des diapositives entrante et sortante, sont masquées et reçoivent un événement hide. Tout de suite après, toutes les diapositives ancêtre de la diapositive entrante, jusqu'à l'ancêtre commun des diapositives entrante et sortante, sont affichées et reçoivent un événement reveal.

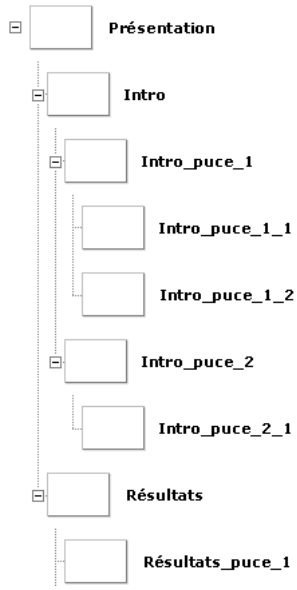
La méthode gotoPreviousSlide() est généralement appelée sur le nœud feuille représentant la diapositive actuelle. Si elle est appelée sur un nœud non feuille, par exemple unNœud, la méthode unNœud.gotoPreviousSlide() permet d'atteindre le premier nœud feuille de la diapositive ou « section » suivante. Pour plus d'informations, consultez l'exemple ci-dessous.

Cette méthode n'a aucun effet lorsqu'elle est appelée sur une diapositive qui ne contient pas la diapositive actuelle (voir l'exemple ci-dessous).

Notez également que cette méthode n'a pas plus d'effet est depuis un gestionnaire d'événement on(hide) ou on(reveal) associé à une diapositive, si ce gestionnaire a été appelé suite à une opération de navigation dans les diapositives.

Exemple

Supposons que, dans la hiérarchie de diapositives suivante, la diapositive nommée `Intro_puce_1_2` soit la diapositive actuellement affichée (c'est-à-dire que `_root.Présentation.currentSlide._name == Intro_puce_1_2`).



Dans ce cas, l'appel de la méthode `Intro_puce_1_2.gotoPreviousSlide()` permet d'atteindre `Intro_puce_1_1`, qui est le frère précédent de `Intro_puce_1_2`.

Cependant, l'appel de `Intro_puce_2.gotoPreviousSlide()` permet d'atteindre `Intro_puce_1_1.`, la première diapositive feuille contenue dans `Intro_puce_1`, qui est le frère suivant de `Intro_puce_2`. De même, l'appel de `Résultats.gotoPreviousSlide()` permet d'atteindre `Intro_puce_1`, la première diapositive feuille contenue dans la diapositive `Intro`.

Enfin, si la diapositive actuelle est `Intro_puce_1_1`, alors l'appel de `Résultats.gotoPreviousSlide()` n'a aucun effet, puisque `Résultats` ne contient pas la diapositive actuelle (ce qui signifie que `Résultats.currentSlide` est `null`).

Voir aussi

`Slide.currentSlide`, `Slide.gotoNextSlide()`, `Slide.previousSlide`

Slide.hideChild

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

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

Description

Evénement : diffusé à chaque fois que l'enfant d'un objet de diapositive passe de l'état visible à l'état invisible. Cet événement est uniquement diffusé par les objets de diapositive, et non par les objets de formulaire. L'événement `hideChild` sert essentiellement à appliquer des transitions « en sortie » à tous les enfants d'une diapositive donnée.

Exemple

Lorsqu'il est associé à la diapositive racine (par exemple, la diapositive Présentation), ce code affiche le nom de chacune des diapositives enfant appartenant à la diapositive racine, au fur et à mesure de leur apparition.

```
on(revealChild) {
  var enfant = objEvt.target._name;
  trace(enfant + " vient de s'afficher");
}
```

Voir aussi

`Slide.revealChild`

Slide.indexInParentSlide

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`maDiapo``.indexInParent`

Description

Propriété (lecture seule) : renvoie l'index de *`maDiapo`* dans la liste de diapositives enfant de son parent ; la numérotation de l'index commence à zéro.

Exemple

Le code suivant utilise les propriétés `indexInParent` et `Slide.numChildSlides` pour afficher l'index de la diapositive actuellement affichée ainsi que le nombre total de diapositives contenues dans la diapositive parent. Pour utiliser ce code, associez-le à une diapositive parent contenant une ou plusieurs diapositives enfant.

```
on (revealChild) {
  trace("Affichage de la diapositive "+(currentSlide.indexInParentSlide+1)+"
  sur "+currentSlide._parent.numChildSlides);
}
```

Notez que, du fait que cette propriété est basée sur zéro, sa valeur est incrémentée d'un point (`currentSlide.indexInParent+1`) afin d'afficher des valeurs plus significatives.

Voir aussi

`Slide.numChildSlides`, `Slide.revealChild`

Slide.lastSlide

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`maDiapo``.lastSlide`

Description

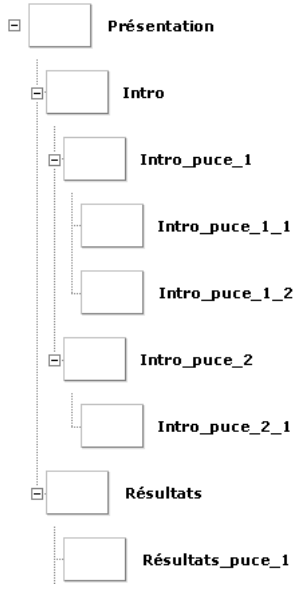
Propriété (lecture seule) : renvoie la dernière diapositive enfant de *`maDiapo`* n'ayant pas d'enfant.

Exemple

Toutes les affirmations suivantes, concernant la hiérarchie de diapositives ci-dessous, sont `vraies` :

```
Présentation.lastSlide._name == Résultats_puce_1;
Intro.lastSlide._name == Intro_puce_1_2;
```

```
Intro_puce_1.lastSlide._name == Intro_puce_1_2;
Résultats.lastSlide._name = Résultats_puce_1;
```



Slide.nextSlide

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

maDiapo.nextSlide

Description

Propriété (lecture seule) : renvoie la diapositive que vous atteindriez en appelant *maDiapo*.gotoNextSlide(), sans effectuer le déplacement. Vous pouvez par exemple utiliser cette propriété pour afficher le nom de la diapositive suivante du diaporama et permettre aux utilisateurs de décider de l'afficher ou non.

Exemple

Dans cet exemple, l'étiquette d'un composant Button nommé `boutonSuivant` affiche le nom de la diapositive suivante du diaporama. S'il n'y a pas de diapositive suivante (si `maDiapo.nextSlide` est `null`), l'étiquette du bouton est mise à jour pour indiquer à l'utilisateur qu'il est à la fin du diaporama.

```
if (maDiapo.nextSlide != null) {
     boutonSuivant.label = "Diapositive suivante : " + maDiapo.nextSlide._name
  + " > ";
} else {
  boutonSuivant.label = "Fin du diaporama.";
}
```

Voir aussi

`Slide.gotoNextSlide()`, `Slide.previousSlide`

Slide.numChildSlides

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`maDiapo``.numChildSlides`

Description

Propriété (lecture seule) : renvoie le nombre de diapositives enfant contenues dans *`maDiapo`*. Notez qu'une diapositive peut contenir des formulaires ou d'autres diapositives. Si *`maDiapo`* contient à la fois des diapositives et des formulaires, cette propriété renvoie uniquement le nombre de diapositives et ne compte pas les formulaires.

Exemple

Cet exemple utilise la propriété Slide.numChildSlide et la méthode `Slide.getChildSlide()` pour itérer sur toutes les diapositives enfant de la diapositive Présentation racine et afficher leurs noms dans le panneau de sortie.

```
var nombreDiapos = _root.Présentation.numChildSlides;
for(var indexDiapo=0; indexDiapo < nombreDiapos; indexDiapo++) {
  var diapoEnfant = _root.Présentation.getChildSlide(indexDiapo);
  trace(diapoEnfant._name);
}
```

Voir aussi

`Slide.getChildSlide()`

Slide.overlayChildren

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

maDiapo.overlayChildren

Description

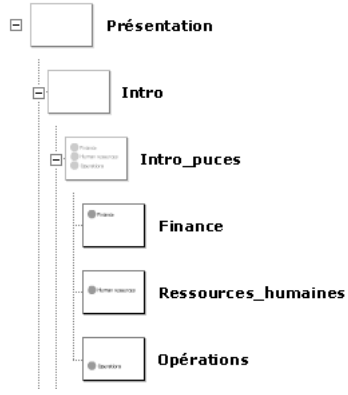
Propriété : détermine si les diapositives enfant de *maDiapo* restent visibles lors de la navigation d'une diapositive enfant à la suivante. Lorsqu'elle est définie sur `true`, la diapositive précédente reste visible lorsque le contrôle passe au frère suivant ; lorsqu'elle est définie sur `false`, la diapositive précédente est invisible lorsque le contrôle passe au frère suivant.

La définition de cette propriété sur true peut s'avérer utile, par exemple, lorsqu'une diapositive donnée contient plusieurs diapositives enfant de type « puce », révélées les unes après les autres (éventuellement à l'aide de transitions) et devant rester visibles lorsque les diapositives frère suivants s'affichent.

Remarque : Cette propriété s'applique uniquement aux descendants immédiats de *maDiapo*, et non pas à toutes les diapositives enfant imbriquées.

Exemple

Par exemple, la diapositive Intro_puces de l'illustration suivante contient trois diapositives enfant (Finance, Ressources humaines et Opérations), affichant chacune une puce distincte. Si vous définissez `Intro_puces.overlayChildren` sur true, toutes les diapositives de type puce restent sur la scène lorsque de nouvelles puces apparaissent.



Slide.parentIsSlide

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

maDiapo.parentIsSlide

Description

Propriété (lecture seule) : valeur booléenne (`true` ou `false`) indiquant si l'objet parent de *maDiapo* est également une diapositive. Si l'objet parent de *maDiapo* est une diapositive, ou une sous-classe de la classe Slide, cette propriété renvoie la valeur `true` ; dans le cas contraire, elle renvoie la valeur false.

Si *maDiapo* est la diapositive racine d'un diaporama, cette propriété renvoie `false`, car le parent de la diapositive Présentation est le scénario principal (_level0) et non une diapositive. Elle renvoie également `false` si le parent de *maDiapo* est un formulaire.

Exemple

Le code suivant détermine si l'objet parent de la diapositive `maDiapo` est lui-même une diapositive. Si c'est le cas, `maDiapo` est supposée être la diapositive racine, ou maître, du diaporama et ne pas posséder de frères. Si `maDiapo.parentIsSlide` renvoie `true`, le nombre de frères de `maDiapo` apparaît dans le panneau de sortie.

```
if (maDiapo.parentIsSlide) {
  trace("J'ai " + maDiapo._parent.numChildSlides+" frères");
} else {
  trace("Je suis la diapositive racine et je n'ai pas de frères");
}
```

Voir aussi

Slide.numChildSlides

Slide.playHidden

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

maDiapo.playHidden

Description

Propriété : valeur booléenne spécifiant si la lecture de *maDiapo* doit continuer lorsqu'elle est masquée. Lorsque cette propriété est définie sur `true`, la lecture de *maDiapo* continue même lorsque celle-ci est masquée. Lorsque cette propriété est définie sur `false`, la lecture de *maDiapo* s'arrête dès que celle-ci est masquée ; la lecture reprend à l'image 1 de *maDiapo* dès que celle-ci est affichée de nouveau.

Vous pouvez également définir cette propriété dans l'inspecteur des propriétés, dans l'environnement auteur de Flash.

Slide.previousSlide

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

maDiapo.previousSlide

Description

Propriété (lecture seule) : renvoie la diapositive que vous atteindriez en appelant *maDiapo*.gotoPreviousSlide(), sans effectuer le déplacement. Vous pouvez par exemple utiliser cette propriété pour afficher le nom de la diapositive précédente du diaporama et permettre aux utilisateurs de décider de l'afficher ou non.

Exemple

Dans cet exemple, l'étiquette d'un composant Button nommé `boutonPrécédent` affiche le nom de la diapositive précédente du diaporama. S'il n'y a pas de diapositive précédente (si `maDiapo.previousSlide` est `null`), l'étiquette du bouton est mise à jour pour indiquer à l'utilisateur qu'il est au début du diaporama.

```
if (maDiapo.previousSlide != null) {
     boutonPrécédent.label = "Diapositive précédente : " +
  maDiapo.previous._name + " > ";
} else {
  boutonPrécédent.label = "Vous êtes au début du diaporama.";
```

Voir aussi

Slide.gotoPreviousSlide(), Slide.nextSlide

Slide.revealChild

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

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

Description

Evénement : diffusé à chaque fois que l'enfant d'un objet de diapositive passe de l'état invisible à l'état visible. Cet événement sert essentiellement à appliquer des transitions « en entrée » à toutes les diapositives enfant d'une diapositive donnée.

Exemple

Lorsqu'il est associé à la diapositive racine (la diapositive Présentation, par exemple), ce code affiche le nom de chacune des diapositives enfant, au fur et à mesure de leur apparition.

```
on(revealChild) {
  var enfant = objEvt.target._name;
  trace(enfant + " vient de s'afficher");
}
```

Voir aussi

`Slide.hideChild`

Slide.rootSlide

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

`maDiapo``.rootSlide`

Description

Propriété (lecture seule) : renvoie la diapositive racine de l'arborescence de diapositives ou de la sous-arborescence de diapositives contenant *`maDiapo`*.

Exemple

Supposons que l'une de vos diapositives contienne un clip qui, lorsque l'utilisateur clique dessus, mène à la première diapositive du diaporama. Pour obtenir ce résultat, vous devez associer le code suivant au clip :

```
on(press) {
  _parent.rootSlide.gotoFirstSlide();
}
```

Dans ce cas, `_parent` fait référence à la diapositive contenant l'objet clip.

Classe StyleManager

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 version 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 version 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 version 79.

Edition

Flash MX 2004.

Usage

```
StyleManager.registerInheritingStyle(nomPropriété)
```

Paramètres

nomPropriété 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");
```

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 27 ou *Classe FocusManager*, page 287.

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 545. 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 en 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 programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez `UIObject.setSize()` 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 29.

Un composant TextArea prend en charge les styles suivants :

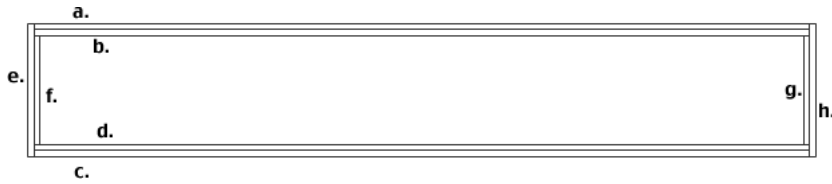
Style	Description
`color`	Couleur par défaut pour le texte.
`embedFonts`	Polices à incorporer dans le document.
`fontFamily`	Nom de police pour du texte.
`fontSize`	Taille en points pour la police.
`fontStyle`	Style de la police, "normal" ou "italic".
`fontWeight`	Epaisseur de la police, "normal" ou "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	Lettre
`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 TextArea

Héritage UIObject > UIComponent > View > ScrollView > TextArea

Nom de classe ActionScript mx.controls.TextArea

Les propriétés de la classe TextArea vous permettent de définir le contenu, le formatage et la position horizontale et verticale du texte 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"`).
`TextArea.html`	Un indicateur qui indique si le champ de texte peut être formaté en 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"`).
`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 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(change){
  ...
}
```

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.change = function(objetEvt){
  ...
}
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 (également appelée *gestionnaire*) sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

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);
}
```

Voir aussi

`UIEventDispatcher.addEventListener()`

TextArea.editable

Disponibilité

Flash Player 6 version 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 version 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 version 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 version 79.

Edition

Flash MX 2004.

Usage

OccurrenceDeZoneDeTexte.html

Description

Propriété : valeur booléenne indiquant si le champ de texte est formaté en 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 version 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 version 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 version 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;
```

Voir aussi

`TextArea.vPosition`

TextArea.maxVPosition

Disponibilité

Flash Player 6 version 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;
```

Voir aussi

`TextArea.hPosition`

TextArea.password

Disponibilité

Flash Player 6 version 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 version 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 version 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 version 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 version 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 version 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 en 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 27 ou *Classe FocusManager*, page 287.

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 533. 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 programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez `UIObject.setSize()` 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".

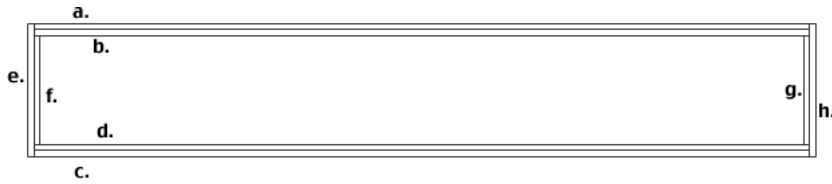
Utilisation d'enveloppes avec le composant TextInput

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	Lettre
`borderColor`	a
`highlightColor`	b

Styles RectBorder	Lettre
borderColor	c
shadowColor	d
borderCapColor	e
shadowCapColor	f
shadowCapColor	g
borderCapColor	h

Les propriétés de style définissent les positions suivantes sur la bordure :



Classe TextInput

Héritage UIObject > UIComponent > TextInput

Nom de classe ActionScript mx.controls.TextInput

Les propriétés de la classe TextInput vous permettent de définir le contenu, le formatage et la position horizontale du texte 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 287.

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 des classes *UIObject* et *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 des classes *UIObject* et *UIComponent*.

Evénements de la classe TextInput

Evénement	Description
`TextInput.change`	Diffusé lorsque le champ de saisie est modifié.
`TextInput.enter`	Diffusé lorsque la touche Entrée est enfoncée.

Hérite de toutes les méthodes des classes *UIObject* et *UIComponent*.

TextInput.change

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(change){
   ...
}
```

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.change = function(objetEvt){
  ...
}
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 (dans ce cas, `change`) qui est géré par une fonction (également appelée *gestionnaire*) sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

Exemple

Cet exemple définit un indicateur dans l'application qui indique si le contenu du champ TextInput a été modifié :

```
form.change = function(objEvt){
  // objEvt.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);
```

Voir aussi

`UIEventDispatcher.addEventListener()`

TextInput.editable

Disponibilité

Flash Player 6 version 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 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(enter){
  ...
}
```

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.enter = function(objetEvt){
  ...
}
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 (dans ce cas, `enter`) qui est géré par une fonction (également appelée *gestionnaire*) sur 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 transmet automatiquement un objet événement (*objetEvt*) 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 événement, consultez *Objets événement*, page 592.

Exemple

Cet exemple définit un indicateur 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);
```

Voir aussi

`UIEventDispatcher.addEventListener()`

TextInput.hPosition

Disponibilité

Flash Player 6 version 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 version 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 `longueur` :

```
var longueur = maSaisieDeTexte.length;
```

TextInput.maxChars

Disponibilité

Flash Player 6 version 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 version 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 version 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 version 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 version 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"
```

Interface TransferObject

Nom de classe ActionScript mx.data.to.TransferObject

L'interface TransferObject définit un ensemble de méthodes que les éléments gérés par le composant DataSet doivent implémenter. La propriété `DataSet.itemClassName` précise le nom de la classe d'objets de transfert qui sera instanciée à chaque fois qu'un nouvel élément est nécessaire. Vous pouvez également spécifier cette propriété pour un composant DataSet sélectionné à l'aide de l'inspecteur des propriétés.

Méthodes de l'interface TransferObject

Méthode	Description
`TransferObject.clone()`	Crée une nouvelle occurrence de l'objet de transfert.
`TransferObject.getPropertyData()`	Renvoie les données de l'objet de transfert.
`TransferObject.setPropertyData()`	Définit les données de l'objet de transfert.

TransferObject.clone()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
class ClasseDeLobjet implements mx.data.to.TransferObject
  function clone() {
    // votre code ici
  }
}
```

Renvoie

Copie de l'objet de transfert.

Description

Méthode : crée une occurrence de l'objet de transfert. L'implémentation de cette méthode permet de créer une copie de l'objet de transfert existant et de ses propriétés, puis de renvoyer cet objet.

Exemple

```
class ClasseDeLobjet implements mx.data.to.TransferObject {
  function clone():Object {
    var b:ContactClass = new ContactClass();
    for (var p in this) {
      b[p] = this[p];
    }
    return b;
  }
}
```

TransferObject.getPropertyData()

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

```
class ClasseDeLobjet implements mx.data.to.TransferObject {
  function getPropertyData() {
  // votre code ici
  }
}
```

Renvoie

Un objet.

Description

Méthode : renvoie les données de cet objet de transfert. L'implémentation de cette méthode peut permettre de renvoyer un objet ActionScript anonyme avec les propriétés et valeurs correspondantes.

Exemple

```
class Contact implements mx.data.to.TransferObject {
function getPropertyData():Object {
  var internalData:Object = { name:name, readOnly:_readOnly, phone:phone,
  zip:zip.zipPlus4 };
  return( internalData );
}
```

TransferObject.setPropertyData()

Disponibilité

Flash Player 7.

Edition

Flash MX 2004.

Utilisation

```
class votreClasse implements TransferObject {
  function setPropertyData(DonnéesProp) {
  // votre code ici
  }
}
```

Paramètres

DonnéesProp Objet contenant les données affectées à l'objet de transfert.

Renvoie

Rien.

Description

Méthode : définit les données de l'objet de transfert. Le paramètre *DonnéesProp* est un objet dont les champs contiennent les données affectées par le composant DataSet à l'objet de transfert.

Exemple

```
class Contact implements mx.data.to.TransferObject {

  function setPropertyData( data: Object ) : Void {
    _readOnly = data.readOnly;
    téléphone = data.phone;
    codePostal = new mx.data.types.ZipCode( data.zip );
  }

public var nom:String;
  public var téléphone:String;
  public var codePostal:ZipCode;
  private var _readOnly:Boolean; // indique si inaltérable
}
```

Composant Tree (Flash Professionnel uniquement)

Le composant Tree permet à l'utilisateur d'afficher des données hiérarchiques. L'arborescence apparaît dans une zone (une occurrence de composant List, par exemple) ; chaque élément de l'arborescence, appelé *nœud*, peut être une *feuille* ou une *branche*. Par défaut, une feuille est représentée par une étiquette de texte placée à côté d'une icône de fichier, tandis qu'une branche est représentée par une étiquette de texte placée à côté d'une icône de dossier, avec un triangle d'affichage que l'utilisateur peut ouvrir pour afficher les enfants. Les enfants d'une branche peuvent être eux-mêmes des feuilles ou des branches.

Les données d'un composant Tree doivent être fournies à partir d'une source de données XML. Pour plus d'informations, consultez la section suivante.

Lorsqu'une occurrence Tree a le focus (après que l'utilisateur a cliqué ou utilisé la touche de tabulation), il est possible d'utiliser les touches suivantes pour la contrôler :

Touche	Description
Flèche vers le bas	Déplace la sélection vers le bas.
Flèche vers le haut	Déplace la sélection vers le haut.
Flèche vers la droite	Ouvre un nœud de branche sélectionné. Si une branche est déjà ouverte, se déplace vers le premier nœud enfant.
Flèche vers la gauche	Ferme un nœud de branche sélectionné. Lorsque vous vous trouvez sur une feuille d'un nœud de branche fermé, se déplace vers le nœud parent.
Espace	Ouvre ou ferme un nœud de branche sélectionné.
Fin	Déplace la sélection en bas de la liste.
Origine	Déplace la sélection en haut de la liste.
Pg. Suiv.	Déplace la sélection d'une page vers le bas.
Pg. Préc.	Déplace la sélection d'une page vers le haut.
Contrôle	Permet plusieurs sélections non contiguës.
Maj	Permet plusieurs sélections contiguës.

Le composant Tree n'est pas accessible dans les logiciels de lecture d'écran.

Utilisation du composant Tree (Flash Professionnel uniquement)

Le composant Tree permet de représenter des données hiérarchiques, telles que des dossiers d'adresses électroniques de clients, des fenêtres d'exploration de fichiers ou des systèmes de navigation par catégorie destinés aux inventaires. Le plus souvent, les données d'une arborescence sont récupérées à partir d'un serveur sous forme de code XML, mais il peut également s'agir de code XML créé dans l'environnement de programmation de Flash. Le meilleur moyen de créer du code XML pour l'arborescence consiste à utiliser l'*Interface TreeDataProvider (Flash Professionnel uniquement)*. Vous pouvez aussi utiliser la classe XML d'ActionScript ou créer une chaîne XML. Une fois votre source de données XML créée (ou chargée à partir d'une source externe), vous l'affectez à la propriété *Tree.dataProvider*.

Le composant Tree comprend deux ensembles d'API : la classe Tree et l'interface TreeDataProvider. La classe Tree contient les méthodes et propriétés de configuration visuelle. L'interface TreeDataProvider vous permet de construire votre code XML pour l'ajouter à plusieurs occurrences de composants Tree. Un objet TreeDataProvider diffuse des modifications vers l'ensemble des arborescences (occurrences de composant Tree) qui l'utilisent. De même, tout objet XML ou XMLNode existant sur la même image en tant qu'arborescence ou menu reçoit automatiquement les méthodes et propriétés TreeDataProvider. Pour plus d'informations, consultez *Interface TreeDataProvider (Flash Professionnel uniquement)*, page 577.

Formatage du code XML pour le composant Tree

Le composant Tree est conçu pour afficher des structures de données hiérarchiques. XML est le modèle de données du composant Tree. Il est important de comprendre la relation liant la source de données XML au composant Tree.

Examinez l'exemple de source de données XML suivant :

```
<node>
  <node label = "Courrier">
       <node label = "BOITE DE RECEPTION"/>
       <node label = "Dossier personnel">
           <node label = "Business" isBranch="true" />
           <node label ="Demo" isBranch="true" />
           <node label ="Personnel" isBranch="true" />
           <node label ="Courrier enregistré" isBranch="true" />
           <node label ="barre" isBranch="true" />
       </node>
       <node label ="Envoyé" isBranch="true" />
       <node label ="Corbeille"/>
  </node>
</node>
```

Remarque : L'attribut `isBranch` est en lecture seule ; vous ne pouvez pas le configurer directement. Pour cela, appelez la méthode `Tree.setIsBranch()`.

Les nœuds de la source de données XML peuvent avoir n'importe quel nom. Notez dans l'exemple ci-dessus que chaque nœud porte l'appellation générique « node ». L'arborescence lit le fichier XML et construit la l'affichage de la structure hiérarchique en respectant la relation imbriquée des nœuds.

Un nœud XML peut être de l'un des deux types suivants : branche ou feuille. Une branche peut contenir plusieurs nœuds enfants ; elle prend la forme d'une icône de dossier, avec un triangle de divulgation permettant à l'utilisateur d'ouvrir et de refermer le dossier. Une feuille prend la forme d'une icône de fichier ; elle ne peut pas contenir de nœuds enfants. Les feuilles et les branches peuvent être des racines ; les nœuds de racine apparaissent en haut de l'arborescence et n'ont aucun parent. Les icônes sont personnalisables ; pour plus d'informations, consultez *Utilisation d'enveloppes avec le composant Tree*, page 564.

Il existe plusieurs moyens de structurer le fichier XML. Le composant Tree n'est pas conçu pour utiliser tous les types de structures XML, il est donc important d'utiliser un code XML qu'il peut interpréter. N'imbriquez pas des attributs de nœud dans un nœud enfant ; chaque nœud doit contenir l'ensemble des attributs qui lui sont nécessaires. De même, les attributs de chaque nœud doivent être cohérents. Par exemple, pour décrire une structure de boîte de réception à l'aide d'un composant Tree, veillez à utiliser les mêmes attributs sur chaque nœud (message, données, heure, pièces jointes, etc.). Cela permet de définir les différents éléments affichables ; vous pouvez explorer la hiérarchie pour comparer les données.

Lorsqu'une arborescence affiche un nœud, elle utilise l'attribut `label` du nœud par défaut comme étiquette de texte. S'il existe d'autres attributs, ils deviennent des propriétés complémentaires des attributs du nœud dans l'arborescence.

Le nœud racine réel est interprété comme étant le composant Tree lui-même. Cela signifie que l'objet `firstChild` (le premier enfant de l'arborescence, soit `<node label="Courrier">`, dans l'exemple) devient le nœud racine dans l'arborescence. Ainsi une arborescence peut avoir plusieurs dossiers racines. Dans cet exemple, un seul dossier racine est affiché dans l'arborescence : "Courrier". En revanche, si vous ajoutez des nœuds frère à ce niveau dans le fichier XML, plusieurs nœuds racines s'afficheront dans l'arborescence.

Paramètres des occurrences de composant Tree

Vous pouvez définir les paramètres de programmation suivants pour chaque occurrence de composant Tree dans l'inspecteur des propriétés ou le panneau Inspecteur des composants :

multipleSelection Valeur Booléenne indiquant si l'utilisateur peut sélectionner plusieurs éléments (`true`) ou non (`false`). La valeur par défaut est false.

rowHeight Hauteur en pixels de chaque ligne. La valeur par défaut est 20.

Vous pouvez rédiger du code ActionScript pour contrôler ces options et des options complémentaires pour le composant Tree à l'aide de ses propriétés, méthodes et événements. Pour plus d'informations, consultez *Classe Tree (Flash Professionnel uniquement)*, page 565.

Il n'est pas possible d'entrer des paramètres de données dans l'inspecteur des propriétés ou dans le panneau Inspecteur de composants du composant Tree comme vous le faites pour les autres composants. Pour plus d'informations, consultez *Utilisation du composant Tree (Flash Professionnel uniquement)*, page 559 et *Création d'une application à l'aide du composant Tree*, page 561.

Création d'une application à l'aide du composant Tree

Dans cet exemple, un développeur crée une application de courrier électronique et choisit d'utiliser un composant Tree pour afficher les boîtes de réception.

Vous ne pouvez pas entrer des paramètres de données dans l'inspecteur des propriétés ou dans le panneau Inspecteur de composants comme vous le faites pour les autres composants. Dans la mesure où la structure des données des composants Tree est plus complexe, vous devez soit importer un objet XML lors de l'exécution, soit en construire un dans Flash au cours de la programmation. Pour créer un objet XML dans Flash, vous pouvez utiliser l'interface TreeDataProvider, l'objet XML d'ActionScript ou créer une chaîne XML. Chacune de ces options est expliquée dans les procédures suivantes.

Pour ajouter un composant Tree à une application :

1. Dans Flash, choisissez Fichier > Nouveau et sélectionnez Document Flash.
2. Dans le panneau Composants, double-cliquez sur le composant Tree pour l'ajouter dans la scène.
3. Dans l'inspecteur des propriétés, entrez **monArborescence** comme nom d'occurrence.

4 Dans le panneau Actions de l'image 1, entrez le code suivant pour créer un gestionnaire d'événements `change` :

```
objetDécoute = new Object();
objetDécoute.change = function(objetEvénement){
  trace(objetEvénement.target.selectedItem.attributes.label + " a été
  sélectionné") ;
}
monArborescence.addEventListener("change", objetDécoute);
```

L'action `trace` à l'intérieur du gestionnaire envoie un message vers le panneau de sortie à chaque fois qu'un élément de l'arborescence est sélectionné.

5 Effectuez l'une des procédures suivantes pour charger ou créer une source de données XML pour l'arborescence.

Pour charger une source de données XML à partir d'un fichier externe, procédez comme suit :

1 Suivez les étapes ci-dessus pour ajouter un composant Tree dans une application et créer un gestionnaire d'événements `change`.

2 Dans le panneau Actions, sur l'Image 1, entrez le code suivant :

```
FDMonArborescence = new XML();
FDMonArborescence.ignoreWhite = true;
FDMonArborescence.load("http://monServeur.monDomaine.com/source.xml");
FDMonArborescence.onLoad = function(){
  monArborescence.dataProvider = FDMonArborescence;
}
```

Ce code crée un objet XML ActionScript appelé `FDMonArborescence` (Fournisseur de données de mon arborescence) et appelle la méthode `XML.load()` pour charger une source de données XML. Le code définit alors un gestionnaire d'événements `onLoad` qui définit le nouvel objet XML comme la propriété `dataProvider` de l'occurrence `monArborescence` pendant le chargement. Pour plus d'informations sur l'objet XML, consultez le Dictionnaire ActionScript de l'aide.

3 Choisissez Contrôle > Tester l'animation.

Dans le fichier SWF, vous pouvez voir la structure XML qui s'affiche dans l'arborescence. Cliquez sur les éléments de l'arborescence pour voir les actions `trace` dans le gestionnaire d'événements `change` envoyer les données vers le panneau de sortie.

Pour utiliser la classe TreeDataProvider pour créer du code XML dans Flash, effectuez les actions suivantes :

1 Suivez les étapes de la première procédure ci-dessus pour ajouter un composant Tree dans une application et créer un gestionnaire d'événements `change`.

2 Dans le panneau Actions, sur l'Image 1, entrez le code suivant :

```
var FDMonArborescence = new XML();
FDMonArborescence.addTreeNode("Répertoires locaux", 0);

// Utilisez XML.firstChild pour imbriquer les nœuds enfant sous les
  répertoires locaux
var NœudMonArborescence = FDMonArborescence.firstChild;
NœudMonArborescence.addTreeNode("Boîte de réception", 1);
NœudMonArborescence.addTreeNode("Boîte d'envoi", 2);
NœudMonArborescence.addTreeNode("Eléments envoyés", 3);
NœudMonArborescence.addTreeNode("Eléments supprimés", 4);
```

```
// Affectez la source de données FDMonArborescence avec le composant
  monArborescence
monArborescence.dataProvider = FDMonArborescence;

// Définissez chacun des 4 nœuds enfant comme étant des branches
for(var i=0; i<NœudMonArborescence.childNodes.length; i++){
    var node = NœudMonArborescence.getTreeNodeAt(i);
    monArborescence.setIsBranch(node, true);
}
```

Ce code crée un objet XML appelé `FDMonArborescence`. Tout objet XML placé sur la même image qu'un composant Tree reçoit automatiquement l'ensemble des propriétés et des méthodes de l'API TreeDataProvider. La deuxième ligne de code crée un nœud racine simple appelé Répertoires locaux. Pour obtenir des informations détaillées sur le reste du code, consultez les commentaires (lignes précédées du signe `//`) dans le code.

3 Choisissez Contrôle > Tester l'animation.

Dans le fichier SWF, vous pouvez voir la structure XML qui s'affiche dans l'arborescence. Cliquez sur les éléments de l'arborescence pour voir les actions `trace` dans le gestionnaire d'événements `change` envoyer les données vers le panneau de sortie.

Pour utiliser la classe XML d'ActionScript pour créer du code XML, effectuez les opérations suivantes :

1 Suivez les étapes de la première procédure ci-dessus pour ajouter un composant Tree dans une application et créer un gestionnaire d'événements `change`.

2 Dans le panneau Actions, sur l'Image 1, entrez le code suivant :

```
// Créer un objet XML
var FDMonArborescence = new XML();
// Créer des valeurs de nœud
var monNœud0 = FDMonArborescence.createElement("nœud");
monNœud0.attributes.label = "Répertoires locaux";
monNœud0.attributes.data = 0;

var monNœud1 = FDMonArborescence.createElement("nœud");
monNœud1.attributes.label = "Boîte de réception";
monNœud1.attributes.data = 1;

var monNœud2 = FDMonArborescence.createElement("nœud");
monNœud2.attributes.label = "Boîte d'envoi";
monNœud2.attributes.data = 2;

var monNœud3 = FDMonArborescence.createElement("nœud");
monNœud3.attributes.label = "Eléments envoyés";
monNœud3.attributes.data = 3;

var monNœud4 = FDMonArborescence.createElement("nœud");
monNœud4.attributes.label = "Eléments supprimés";
monNœud4.attributes.data = 4;
// Affecter des nœuds dans la hiérarchie de l'arborescence XML
FDMonArborescence.appendChild(monNœud0);
FDMonArborescence.firstChild.appendChild(monNœud1);
FDMonArborescence.firstChild.appendChild(monNœud2);
FDMonArborescence.firstChild.appendChild(monNœud3);
FDMonArborescence.firstChild.appendChild(monNœud4);
// Affecter la source de données FDMonArborescence avec le contrôle
  d'arborescence
monArborescence.dataProvider = FDMonArborescence;
```

Veuillez lire les commentaires du code (lignes commençant par `//`) pour obtenir une description du code. Pour plus d'informations sur l'objet XML, consultez le Dictionnaire ActionScript de l'aide.

3 Choisissez Contrôle > Tester l'animation.

Dans le fichier SWF, vous pouvez voir la structure XML qui s'affiche dans l'arborescence. Cliquez sur les éléments de l'arborescence pour voir les actions `trace` dans le gestionnaire d'événements `change` envoyer les données vers le panneau de sortie.

Pour utiliser une chaîne formée pour créer du code XML dans Flash pendant la programmation, effectuez les opérations suivantes :

1 Suivez les étapes de la première procédure ci-dessus pour ajouter un composant Tree dans une application et créer un gestionnaire d'événements `change`.

2 Dans le panneau Actions, sur l'Image 1, entrez le code suivant :

```
FDMonArborescence = new XML("<node label ='Répertoires'><node label='Boîte
  de réception' data='0'/><node label='Boîte d'envoi' data='1'/></node>");
monArborescence.dataProvider = FDMonArborescence;
```

Le code ci-dessus crée un objet XML `FDMonArborescence` et l'affecte à la propriété `dataProvider` de `monArborescence`.

3 Choisissez Contrôle > Tester l'animation.

Dans le fichier SWF, vous pouvez voir la structure XML qui s'affiche dans l'arborescence. Cliquez sur les éléments de l'arborescence pour voir les actions `trace` dans le gestionnaire d'événements `change` envoyer les données vers le panneau de sortie.

Personnalisation du composant Tree (Flash Professionnel uniquement)

Vous pouvez transformer un composant Tree horizontalement et verticalement au cours de la programmation et à l'exécution. Lors de la programmation, choisissez le composant sur la scène et utilisez l'outil Transformer librement ou une commande de modification > transformation. A l'exécution, utilisez la méthode `setSize()` (consultez `UIObject.setSize()`). Lorsqu'une arborescence n'est pas assez large pour afficher le texte des nœuds, celui-ci apparaît tronqué.

Utilisation de styles avec le composant Tree

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

Utilisation d'enveloppes avec le composant Tree

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

Classe Tree (Flash Professionnel uniquement)

Héritage UIObject > UIComponent > View > ScrollView > ScrollSelectList > List > Tree

Nom de classe Actionscript mx.controls.Tree

Méthodes de la classe Tree

Méthode	Description
`Tree.addTreeNode()`	Ajoute un nœud dans une arborescence.
`Tree.addTreeNodeAt()`	Ajoute un nœud à un endroit spécifique de l'arborescence.
`Tree.getDisplayIndex()`	Renvoie l'index d'affichage d'un nœud donné.
`Tree.getIsBranch()`	Précise si le dossier est une branche (possède une icône de dossier et une flèche d'agrandissement).
`Tree.getIsOpen()`	Indique si une branche est ouverte ou fermée.
`Tree.getNodeDisplayedAt()`	Renvoie l'index d'affichage d'un nœud donné.
`Tree.getTreeNodeAt()`	Renvoie un nœud à la racine de l'arborescence.
`Tree.removeAll()`	Supprime tous les nœuds d'une occurrence d'arborescence et actualise l'arborescence.
`Tree.removeTreeNodeAt()`	Supprime un nœud à un endroit indiqué et actualise l'arborescence.
`Tree.setIsBranch()`	Indique si un nœud est une branche (reçoit une icône de dossier et une flèche d'agrandissement).
`Tree.setIcon()`	Précise si un nœud est ouvert ou fermé.
`Tree.setIsOpen()`	Indique un symbole à utiliser en tant qu'icône d'un nœud.

Hérite de l'ensemble des méthodes des classes UIComponent, UIObject, View, ScrollView, ScrollSelectList et List.

Propriétés de la classe Tree

Propriété	Description
`Tree.dataProvider`	Indique une source de données XML.
`Tree.firstVisibleNode`	Indique le premier nœud en haut de l'affichage.
`Tree.selectedNode`	Indique le nœud sélectionné dans une occurrence d'arborescence.
`Tree.selectedNodes`	Indique les nœuds sélectionnés dans une occurrence d'arborescence.

Hérite de l'ensemble des propriétés des classes UIComponent, UIObject, View, ScrollView, ScrollSelectList et List.

Evénements de la classe Tree

Evénement	Description
`Tree.nodeClose`	Diffusé lorsqu'un nœud est fermé par un utilisateur.
`Tree.nodeOpen`	Diffusé lorsqu'un nœud est ouvert par un utilisateur.

Hérite de l'ensemble des événements des classes UIComponent, UIObject, View, ScrollView, ScrollSelectList et List.

Tree.addTreeNode()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

Usage 1 :

```
monArborescence.addTreeNode(étiquette [, données])
```

Usage 2 :

```
monArborescence.addTreeNode(enfant)
```

Paramètres

étiquette Chaîne qui affiche le nœud, ou objet possédant un champ "étiquette" (ou n'importe quel nom de champ d'étiquette spécifié par la propriété `labelField`).

données Objet de n'importe quel type associé au nœud. Ce paramètre est facultatif.

enfant Tout objet XMLNode.

Renvoie

Nœud XML ajouté.

Description

Méthode : ajoute un nœud enfant à l'arborescence. Le nœud est construit à partir des informations fournies dans les paramètres d'étiquette et de données (Usage 1) ou à partir du nœud enfant (objet XMLNode) prédéfini (Usage 2). Si le nœud ajouté existe déjà, il est supprimé de son emplacement précédent.

Remarque : L'appel de cette méthode permet d'actualiser l'affichage.

Exemple

Le code suivant ajoute un nouveau nœud à la racine de `monArborescence`. La deuxième ligne du code déplace un nœud à partir de la racine de `maDeuxièmeArborescence` vers la racine de `monArborescence` :

```
monArborescence.addTreeNode("Boîte de réception", 3);
monArborescence.addTreeNode(maDeuxièmeArborescence.getTreeNodeAt(3));
```

Tree.addTreeNodeAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

Usage 1 :

```
monArborescence.addTreeNodeAt(index, étiquette [, données])
```

Usage 2 :

```
monArborescence.addTreeNodeAt(index, enfant)
```

Paramètres

index Ordre (parmi les nœuds enfants) dans lequel le nœud doit être ajouté.

étiquette Chaîne qui affiche le nœud.

données Objet de n'importe quel type associé au nœud. Ce paramètre est facultatif.

enfant Tout objet XMLNode.

Renvoie

Nœud XML ajouté.

Description

Méthode : ajoute un nœud à l'endroit spécifié de l'arborescence. Le nœud est construit à partir des informations fournies dans les paramètres d'étiquette et de données (Usage 1) ou à partir de l'objet XMLNode prédéfini (Usage 2). Si le nœud ajouté existe déjà, il est supprimé de son emplacement précédent.

Remarque : L'appel de cette méthode permet d'actualiser l'affichage.

Exemple

L'exemple suivant ajoute un nouveau nœud en tant que deuxième enfant de la racine de `monArborescence`. La deuxième ligne déplace un nœud de `maDeuxièmeArborescence` pour en faire le quatrième enfant à la racine de `monArborescence` :

```
monArborescence.addTreeNodeAt(1, "Boîte de réception", 3);
monArborescence.addTreeNodeAt(3,maDeuxièmeArborescence.getTreeNodeAt(3));
```

Tree.dataProvider

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monArborescence.dataProvider
```

Description

Propriété : la propriété `dataProvider` peut être un objet XML ou une chaîne. Si `dataProvider` est un objet XML, cet objet vient s'ajouter directement dans l'arborescence. Si `dataProvider` est une chaîne, il doit contenir un objet XML valide lu par l'arborescence et converti en objet XML.

Vous pouvez charger l'objet XML à partir d'une source externe à l'exécution ou le créer dans Flash lors de la programmation. Pour créer un objet XML, vous avez la possibilité d'utiliser les méthodes TreeDataProvider ou les méthodes et propriétés de la classe XML intégrée d'ActionScript. Vous pouvez également créer une chaîne contenant un objet XML.

Les objets XML qui se trouvent sur la même image sous forme de composant Tree contiennent automatiquement les méthodes et propriétés `TreeDataProvider`. Vous pouvez utiliser les objets XML ou XMLNode d'ActionScript.

Exemple

L'exemple suivant importe un fichier XML et l'affecte à l'occurrence de composant Tree nommée `monArborescence` :

```
FDMonArborescence = new XML();
FDMonArborescence.ignoreWhite = true;
FDMonArborescence.load("http://monServeur.monDomaine.com/source.xml");
FDMonArborescence.onLoad = function(){
   monArborescence.dataProvider = FDMonArborescence;
}
```

Remarque : La plupart des fichiers XML contiennent des espaces : pour que Flash les ignore, vous devez définir la propriété `ignoreWhite` sur `true`.

Tree.firstVisibleNode

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monArborescence.firstVisibleNode
```

Description

Propriété : le premier nœud en haut de l'affichage. Si le nœud se trouve sous un nœud qui n'est pas développé, configurer `firstVisibleNode` n'a aucun effet. La valeur par défaut correspond au premier nœud visible ou est undefined s'il n'existe aucun nœud visible. La valeur de cette propriété est un objet XMLNode.

Remarque : Sa configuration est similaire à celle de `List.vPosition`.

Exemple

L'exemple suivant permet de définir la position de défilement en haut de l'affichage :

```
monArborescence.firstVisibleNode = monArborescence.getTreeNodeAt(0);
```

Tree.getIsBranch()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monArborescence.getIsBranch(nœud)
```

Paramètres

nœud Objet XMLNode.

Renvoie

Valeur Booléenne qui indique si le nœud est une branche (`true`) ou non (`false`).

Description

Méthode : indique si le nœud spécifié possède une icône de dossier et une flèche d'agrandissement (et est par conséquent une *branche*). Cette configuration est automatique lorsque des enfants sont ajoutés au nœud. Il vous suffit simplement d'appeler la méthode `setIsBranch()` pour créer des dossiers vides. Pour plus d'informations, consultez `Tree.setIsBranch()`.

Exemple

Le code suivant affecte l'état du nœud à une variable :

```
var ouverture = monArborescence.getIsBranch(monArborescence.getTreeNodeAt(1));
```

Voir aussi

`Tree.setIsBranch()`

Tree.getIsOpen()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monArborescence.getIsOpen(nœud)
```

Paramètres

nœud Objet XMLNode.

Renvoie

Valeur Booléenne indiquant si l'arborescence est ouverte (`true`) ou ne l'est pas (`false`).

Description

Méthode : indique si le nœud spécifié est ouvert ou fermé.

Exemple

Le code suivant affecte l'état du nœud à une variable :

```
var ouverture = monArborescence.getIsOpen(monArborescence.getTreeNodeAt(1));
```

Tree.getDisplayIndex()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monArborescence.getDisplayIndex(nœud)
```

Paramètres

nœud Objet XMLNode.

Renvoie

L'index du nœud spécifié ou undefined si le nœud n'est pas actuellement affiché.

Description

Méthode : renvoie l'index d'affichage du nœud spécifié dans le paramètre *nœud*.

L'index d'affichage est un tableau d'éléments pouvant s'afficher dans la fenêtre de l'arborescence. Par exemple, les enfants d'un nœud fermé ne se trouvent pas dans l'index d'affichage. L'index d'affichage commence à 0 et répertorie les éléments visibles, quels que soient leurs parents. En d'autres termes, l'index d'affichage correspond au numéro (commençant par 0) des lignes affichées.

Exemple

Le code suivant permet d'obtenir l'index d'affichage de `monNœud` :

```
var x = monArborescence.getDisplayIndex(monNœud);
```

Tree.getNodeDisplayedAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monArborescence.getNodeDisplayedAt(index)
```

Paramètres

index Nombre entier représentant la position d'affichage dans la zone affichable de l'arborescence. Ce nombre est basé sur zéro ; le nœud en première position est 0, celui en deuxième position est 1 et ainsi de suite.

Renvoie

L'objet XMLNode spécifié.

Description

Méthode : permet d'établir la correspondance entre un index d'affichage de l'arborescence et un nœud affiché. Par exemple, si la cinquième ligne de l'arborescence a affiché un nœud se trouvant à huit niveaux de profondeur dans la hiérarchie, ce nœud est renvoyé par le code `getNodeDisplayedAt(4)`.

L'index d'affichage est un tableau d'éléments pouvant s'afficher dans la fenêtre de l'arborescence. Par exemple, les enfants d'un nœud fermé ne se trouvent pas dans l'index d'affichage. L'index d'affichage commence à 0 et répertorie les éléments visibles, quels que soient leurs parents. En d'autres termes, l'index d'affichage correspond au numéro (commençant par 0) des lignes affichées.

Remarque : Les index d'affichage changent à chaque ouverture ou fermeture de nœud.

Exemple

Le code suivant obtient une référence au nœud XML qui correspond à la deuxième ligne affichée dans `monArborescence` :

```
monArborescence.getNodeDisplayedAt(1) ;
```

Tree.getTreeNodeAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monArborescence.getTreeNodeAt(index)
```

Paramètres

index Numéro d'index d'une arborescence.

Renvoie

Objet XMLNode.

Description

Méthode : renvoie le nœud spécifié sur la racine de `monArborescence`.

Exemple

Le code suivant obtient le deuxième nœud du premier niveau de l'arborescence `monArborescence` :

```
monArborescence.getTreeNodeAt(1);
```

Tree.nodeClose

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

```
objetDécoute = new Object();
objetDécoute.nodeClose = function(objetEvénement){
  // insérez votre code ici
}
occurrenceDeMonArborescence.addEventListener("nodeClose", objetDécoute)
```

Description

Evénement : diffusé vers l'ensemble des écouteurs enregistrés lorsque les nœuds d'un composant Tree sont fermés par un utilisateur.

Les composants V2 utilisent un modèle d'événement dispatcher/écouteur. Le composant Tree diffuse un événement `nodeClose` lorsque l'utilisateur clique sur l'un de ses nœuds pour le fermer et lorsque l'événement est géré par une fonction, aussi appelée *gestionnaire*, associée à un objet d'écoute (`objetDécoute`) que vous créez.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (`objetEvt`) au gestionnaire. 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. L'objet événement `Tree.nodeClose` possède une propriété supplémentaire : `nœud` (nœud XML fermé). Pour plus d'informations sur les objets événement, consultez *Objets événement*, page 592.

Exemple

Dans l'exemple suivant, un gestionnaire `objetDécouteDeMonArborescence` est défini et transmis à la méthode `monArborescence.addEventListener()` en tant que second paramètre. L'objet événement est capturé par le gestionnaire `nodeClose` dans le paramètre `objetEvénement`. Lorsque l'événement `nodeClose` est diffusé, une instruction `trace` est envoyée vers le panneau de sortie, comme suit :

```
objetDécouteDeMonArborescence = new Object();
objetDécouteDeMonArborescence.nodeClose = function(objetEvénement){
  trace(objetEvénement.node + " nœud fermé");
}
monArborescence.addEventListener("nodeClose", objetDécouteDeMonArborescence);
```

Tree.nodeOpen

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDécoute = new Object();
objetDécoute.nodeOpen = function(objetEvénement){
  // insérez votre code ici
}
occurrenceDeMonArborescence.addEventListener("nodeOpen", objetDécoute)
```

Description

Evénement : diffusé vers l'ensemble des objets d'écoute enregistrés lorsqu'un utilisateur ouvre un nœud sur un composant Tree.

Les composants V2 utilisent un modèle d'événement dispatcher/écouteur. Le composant Tree distribue un événement `nodeOpen` lorsque l'utilisateur clique sur un nœud pour l'ouvrir et lorsque l'événement est géré par une fonction, appelée aussi *gestionnaire*, associée à un objet d'écoute (*objetDécoute*) que vous créez. Vous appelez la méthode `addEventListener()` et lui transmettez le nom du gestionnaire en tant que paramètre.

Lorsque l'événement est déclenché, il transmet automatiquement un objet événement (*objetEvt*) au gestionnaire. 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. L'objet événement `Tree.nodeOpen` possède une propriété complémentaire : `nœud` (le nœud XML ouvert). Pour plus d'informations sur les objets événement, consultez *Objets événement*, page 592.

Exemple

Dans l'exemple suivant, un gestionnaire `objetDécouteDeMonArborescence` est défini et transmis à la méthode `monArborescence.addEventListener()` en tant que second paramètre. L'objet événement est capturé par le gestionnaire `nodeOpen` dans le paramètre `objetEvénement`. Lorsque l'événement `nodeOpen` est diffusé, une instruction `trace` est envoyée vers le panneau de sortie, comme suit :

```
objetDécouteDeMonArborescence = new Object();
occurrenceDeMonArborescence.nodeOpen = function(objetEvénement){
  trace(objetEvénement.node + "nœud ouvert");
}
monArborescence.addEventListener("nodeOpen", objetDécouteDeMonArborescence);
```

Tree.removeAll()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monArborescence.removeAll()
```

Paramètres

Aucun.

Renvoie

Rien.

Description

Méthode : supprime tous les nœuds et actualise l'arborescence.

Exemple

Le code suivant vide `monArborescence` :

```
monArborescence.removeAll();
```

Tree.removeTreeNodeAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monArborescence.removeTreeNodeAt(index)
```

Paramètres

`index` Numéro d'index d'un enfant de l'arborescence. Chaque enfant d'une arborescence est associé à un numéro d'index en fonction de son ordre de création ; le premier numéro d'index est 0.

Renvoie

Un objet XMLNode ou undefined s'il existe une erreur.

Description

Méthode : supprime un nœud (spécifié par sa position dans l'index) à la racine de l'arborescence et actualise cette dernière.

Exemple

Le code suivant supprime le quatrième enfant de la racine de l'arborescence `monArborescence` :

```
monArborescence.removeTreeNodeAt(3);
```

Tree.setIsBranch()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

```
monArborescence.setIsBranch(nœud, isBranch)
```

Paramètres

`nœud` Nœud XML.

`isBranch` Valeur Booléenne indiquant si le nœud est une branche (`true`) ou non (`false`).

Renvoie

Rien.

Description

Méthode : indique si le nœud possède une icône de dossier et une flèche d'agrandissement et s'il a ou peut avoir des enfants. Un nœud est automatiquement défini en tant que branche lorsqu'il a des enfants ; il vous suffit d'appeler la méthode `setIsBranch()` pour créer un dossier vide. Vous pouvez également créer des branches n'ayant pas encore d'enfants : cela peut être utile si vous souhaitez par exemple que le chargement de nœuds enfant soit déclenché par l'ouverture d'un dossier par l'utilisateur.

L'appel de la méthode `setIsBranch()` permet d'actualiser tous les affichages.

Exemple

Le code suivant transforme un nœud de `monArborescence` en branche :

```
monArborescence.setIsBranch(monArborescence.getTreeNodeAt(1), true);
```

Tree.setIcon()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Utilisation

```
monArborescence.setIcon(nœud, IDLien[, IDLien2])
```

Paramètres

`nœud` Nœud XML.

`IDLien` Identifiant de liaison d'un symbole à utiliser en tant qu'icône à côté du nœud. Ce paramètre est utilisé pour les nœuds de feuille et pour l'état fermé des nœuds de branche.

`IDLien2` Identifiant de liaison d'un symbole à utiliser en tant qu'icône à côté du nœud. Ce paramètre est utilisé pour l'icône représentant l'état ouvert de nœuds de branche.

Renvoie

Rien.

Description

Méthode : définit une icône pour le nœud spécifié. Cette méthode prend un paramètre (*`IDLien`*) pour les nœuds de feuille et deux paramètres (*`IDLien`* et *`IDLien2`*) pour les branches (icônes fermé et ouvert). Le deuxième paramètre est ignoré pour les nœuds de feuille (non branche) et si un seul paramètre est spécifié pour un nœud de branche, l'icône sert pour les deux états, fermé et ouvert.

Exemple

Le code suivant indique qu'un symbole avec l'identifiant de liaison « imageIcon » doit être utilisé à côté du deuxième nœud de `monArborescence` :

```
monArborescence.setIcon(monArborescence.getTreeNodeAt(1), "imageIcon");
```

Tree.setIsOpen()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monArborescence.setIsOpen(node, isOpen[, noEvent])
```

Paramètres

nœud Nœud XML.

isOpen Valeur Booléenne qui ouvre un nœud (`true`) ou le ferme (`false`).

noEvent Valeur Booléenne qui anime (`true`) ou n'anime pas (`false`) la transition d'ouverture. Ce paramètre est facultatif.

Renvoie

Rien.

Description

Méthode : ouvre ou ferme un nœud.

Exemple

Le code suivant ouvre un nœud de `monArborescence` :

```
monArborescence.setIsOpen(monArborescence.getTreeNodeAt(1), true);
```

Tree.selectedNode

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monArborescence.selectedNode
```

Description

Propriété : spécifie le nœud sélectionné dans une occurrence d'arborescence.

Exemple

L'exemple suivant spécifie le nœud correspondant au premier enfant dans `monArborescence` :

```
monArborescence.selectedNode = monArborescence.getTreeNodeAt(0);
```

Voir aussi

`Tree.selectedNodes`

Tree.selectedNodes

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monArborescence.selectedNodes
```

Description

Propriété : spécifie les nœuds sélectionnés dans une occurrence d'arborescence.

Exemple

L'exemple suivant sélectionne le premier et le troisième nœuds enfant dans `monArborescence` :

```
monArborescence.selectedNodes = [monArborescence.getTreeNodeAt(0),
  monArborescence.getTreeNodeAt(2)];
```

Voir aussi

`Tree.selectedNode`

Interface TreeDataProvider (Flash Professionnel uniquement)

TreeDataProvider est une interface ; elle n'a pas besoin d'être instanciée pour être utilisée. Si une classe Tree est comprise dans un fichier SWF, toutes les occurrences XML de ce fichier contiennent l'API TreeDataProvider. Les nœuds d'une arborescence sont autant d'objets XML contenant l'API TreeDataProvider.

Il est recommandé d'utiliser les méthodes de l'API TreeDataProvider pour créer du code XML pour la propriété `Tree.dataProvider`, car seul l'interface TreeDataProvider diffuse des événements vers les composants Tree qui actualisent l'affichage de l'arborescence. Les méthodes de la classe XML intégrée peuvent être utilisées pour créer du code XML, mais elles ne diffusent pas d'événements permettant d'actualiser l'affichage.

Vous pouvez utiliser les méthodes de l'API TreeDataProvider pour contrôler le modèle et l'affichage des données. Vous pouvez utiliser les méthodes de la classe XML intégrée pour les tâches en lecture seule (parcourir la hiérarchie de l'arborescence, par exemple).

Il est possible de sélectionner la propriété qui contrôle le texte à afficher en spécifiant une propriété `labelField` ou `labelFunction`. Par exemple, le code `monArborescence.labelField = "fred";` renvoie une requête pour la valeur de la propriété `FDMonArborescence.attributes.fred` pour le texte d'affichage.

Méthodes de l'interface TreeDataProvider

Méthode	Description
TreeDataProvider.addTreeNode()	Ajoute un nœud enfant à la fin d'un nœud parent.
TreeDataProvider.addTreeNodeAt()	Ajoute un nœud enfant à un endroit spécifié sur le nœud parent.
TreeDataProvider.getTreeNodeAt()	Renvoie l'enfant spécifié d'un nœud.
TreeDataProvider.removeTreeNode()	Supprime un nœud et tous les descendants du nœud à partir du parent du nœud.
TreeDataProvider.removeTreeNodeAt()	Supprime un nœud et tous les descendants du nœud à partir de la position dans l'index du nœud enfant.

Propriétés de l'interface TreeDataProvider

Propriété	Description
TreeDataProvider.attributes.data	Spécifie les données à associer à un nœud.
TreeDataProvider.attributes.label	Spécifie le texte à afficher à côté d'un nœud.

TreeDataProvider.addTreeNode()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

Usage 1 :

```
Nœud.addTreeNode(étiquette, données)
```

Usage 2 :

```
Nœud.addTreeNode(enfant)
```

Paramètres

étiquette Chaîne qui affiche le nœud.

données Objet de n'importe quel type associé au nœud.

enfant Tout objet XMLNode.

Renvoie

Nœud XML ajouté.

Description

Méthode : ajoute un nœud enfant à la racine de l'arborescence. Le nœud est construit à partir des informations fournies dans les paramètres d'étiquette et de données (Usage 1) ou à partir du nœud enfant (objet XMLNode) prédéfini (Usage 2). Si le nœud ajouté existe déjà, il est supprimé de son emplacement précédent.

L'appel de cette méthode permet d'actualiser l'affichage de l'occurrence de l'arborescence.

Exemple

La première ligne du code dans l'exemple suivant localise le nœud auquel ajouter un enfant. La deuxième ligne ajoute un nouveau nœud à un nœud spécifié, comme suit :

```
var nœudDeMonArborescence = FDMonArborescence.firstChild.firstChild;
nœudDeMonArborescence.addTreeNode("Boîte de réception", 3);
```

Le code suivant déplace un nœud d'une arborescence vers la racine d'une autre arborescence :

```
nœudDeMonArborescence.addTreeNode(maDeuxièmeArborescence.getTreeNodeAt(3));
```

TreeDataProvider.addTreeNodeAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

Usage 1 :

```
Nœud.addTreeNodeAt(index, étiquette, données)
```

Usage 2 :

```
Nœud.addTreeNodeAt(index, enfant)
```

Paramètres

index Nombre entier indiquant la position d'index parmi les nœuds enfant à laquelle le nœud doit être ajouté.

étiquette Chaîne qui affiche le nœud.

données Objet de n'importe quel type associé au nœud.

enfant Tout objet XMLNode.

Renvoie

Nœud XML ajouté.

Description

Méthode : ajoute un nœud enfant à l'endroit spécifié dans le nœud parent. Le nœud est construit à partir des informations fournies dans les paramètres d'étiquette et de données (Usage 1) ou à partir du nœud enfant (objet XMLNode) prédéfini (Usage 2). Si le nœud ajouté existe déjà, il est supprimé de son emplacement précédent.

L'appel de cette méthode permet d'actualiser l'affichage de l'occurrence de l'arborescence.

Exemple

Le code suivant localise le nœud auquel vous ajoutez un nœud et ajoute un nouveau nœud en tant que deuxième enfant de la racine :

```
var nœudDeMonArborescence = FDMonArborescence.firstChild.firstChild;
nœudDeMonArborescence.addTreeNodeAt(1, "Boîte de réception", 3);
```

Le code suivant déplace un nœud à partir d'une arborescence pour qu'il devienne le quatrième enfant de la racine d'une autre arborescence :

```
nœudDeMonArborescence.addTreeNodeAt(3,
  maDeuxièmeArborescence.getTreeNodeAt(3));
```

TreeDataProvider.attributes.data

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

Nœud.attributes.data

Description

Propriété : spécifie les données à associer au nœud. Ceci permet d'ajouter la valeur en tant qu'attribut à l'intérieur de l'objet nœud XML. La définition de cette propriété n'actualise pas l'affichage. Cette propriété peut être de n'importe quel type de données.

Exemple

Le code suivant localise le nœud à régler et définit sa propriété `data` :

```
var nœudDeMonArborescence = FDMonArborescence.firstChild.firstChild;
nœudDeMonArborescence.attributes.data = "hi"; // results in <node data =
  "hi">;
```

Voir aussi

TreeDataProvider.attributes.label

TreeDataProvider.attributes.label

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

Nœud.attributes.label

Description

Propriété : chaîne indiquant le texte affiché pour le nœud. Il est affiché dans un attribut de l'objet nœud XML. La définition de cette propriété ne permet pas d'actualiser l'affichage de l'arborescence.

Exemple

Le code suivant localise le nœud à régler et définit sa propriété `label`. Le résultat du code suivant est "`<node label="Courrier">`":

```
var nœudDeMonArborescence = FDMonArborescence.firstChild.firstChild;
nœudDeMonArborescence.attributes.label = "Courrier";
```

Voir aussi

`TreeDataProvider.attributes.data`

TreeDataProvider.getTreeNodeAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

`Nœud``.getTreeNodeAt(`*`index`*`)`

Paramètres

`index` Nombre entier représentant la position du nœud enfant dans le nœud actuel.

Renvoie

Le nœud spécifié.

Description

Méthode : renvoie le nœud enfant spécifié du nœud.

Exemple

Le code suivant localise un nœud, puis obtient le deuxième enfant de `nœudDeMonArborescence` :

```
var nœudDeMonArborescence = FDMonArborescence.firstChild.firstChild;
nœudDeMonArborescence.getTreeNodeAt(1);
```

TreeDataProvider.removeTreeNode()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
Nœud.removeTreeNode()
```

Paramètres

Aucun.

Renvoie

Le nœud XML supprimé ou undefined en cas d'erreur.

Description

Méthode : supprime le nœud spécifié et ses éventuels descendants, à partir de son parent.

Exemple

Le code suivant supprime un nœud :

```
FDMonArborescence.firstChild.removeTreeNode();
```

TreeDataProvider.removeTreeNodeAt()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
Nœud.removeTreeNodeAt(index)
```

Paramètres

index Nombre entier indiquant la position du nœud à supprimer.

Renvoie

Le nœud XML supprimé ou undefined en cas d'erreur.

Description

Méthode : supprime un nœud (et tous les descendants) spécifié par le nœud actuel et la position d'index du nœud enfant. L'appel de cette méthode permet d'actualiser l'affichage.

Exemple

Le code suivant supprime le quatrième enfant d'un nœud spécifié :

```
FDMonArborescence.firstChild.removeTreeNodeAt(3);
```

UIComponent

Héritage UIObject > UIComponent

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 *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 *UIObject*.

Evénements de la classe UIComponent

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

Hérite de tous les événements de la classe *UIObject*.

UIComponent.focusIn

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

```
on(focusIn){
  ...
}
objetDécoute = new Object();
objetDécoute.focusIn = function(objetEvt){
  ...
}
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 (dans ce cas, `focusIn`) qui est géré par une fonction (appelée aussi *gestionnaire*) sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

Exemple

Le code suivant désactive un bouton lorsque l'utilisateur renseigne le champ de texte `txt` :

```
txtListener.handleEvent = function(objetEvt){
  form.button.enabled = false;
}
txt.addEventListener("focusIn", txtListener);
```

Voir aussi

`UIEventDispatcher.addEventListener()`

UIComponent.focusOut

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

```
on(focusOut){
  ...
}
objetDécoute = new Object();
objetDécoute.focusOut = function(objetEvt){
  ...
}
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 (dans ce cas, focusOut) qui est géré par une fonction (appelée aussi *gestionnaire*) sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

Exemple

Le code suivant active un bouton lorsque l'utilisateur quitte le champ de texte `txt` :

```
txtListener.handleEvent = function(objetEvt){
  if (objetEvt.type == focusOut){
      form.button.enabled = true;
  }
}
txt.addEventListener("focusOut", txtListener);
```

Voir aussi

`UIEventDispatcher.addEventListener()`

UIComponent.enabled

Disponibilité

Flash Player 6 version 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 version 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 version 79.

Edition

Flash MX 2004.

Usage

```
on(keyDown){
   ...
}
objetDécoute = new Object();
objetDécoute.keyDown = function(objetEvt){
   ...
}
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 (dans ce cas, keyDown) qui est géré par une fonction (appelée aussi *gestionnaire*) sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

Exemple

Le code suivant fait clignoter une icône lorsque l'utilisateur appuie sur une touche :

```
formListener.handleEvent = function(objEvt)
{
  form.icon.visible = !form.icon.visible;
}
form.addEventListener("keyDown", formListener);
```

UIComponent.keyUp

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

```
on(keyUp){
   ...
}
objetDécoute = new Object();
objetDécoute.keyUp = function(objetEvt){
   ...
}
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 (dans ce cas, keyUp) qui est géré par une fonction (appelée aussi *gestionnaire*) sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

Exemple

Le code suivant fait clignoter une icône lorsqu'une touche est relâchée :

```
formListener.handleEvent = function(objEvt)
{
  form.icon.visible = !form.icon.visible;
}
form.addEventListener("keyUp", formListener);
```

UIComponent.setFocus()

Disponibilité

Flash Player 6 version 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 version 79.

Edition

Flash MX 2004.

Usage

```
occurrence.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;
```

UIEventDispatcher

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 24.

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 version 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 la fonction est averti(e). Vous pouvez appeler cette méthode à partir de toute occurrence de composant. Le code suivant, par exemple, enregistre un écouteur dans l'occurrence 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 25.

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);
```

L'ordre d'exécution n'est pas garanti. Vous ne pouvez pas anticiper l'ordre dans lequel les écouteurs vont être appelés.

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 592.

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.
target	Référence à l'occurrence de composant qui émet l'événement.

UIObject

Héritage MovieClip > UIObject

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. L'habillage de la classe MovieClip permet à Macromedia d'ajouter de nouveaux événements et par la suite d'étendre la fonctionnalité sans perte de contenu. L'habillage de la classe MovieClip permet également aux utilisateurs qui n'ont pas l'habitude des concepts d'animation et d'image propres à Flash d'utiliser l'API pour créer des applications basées sur les composants sans avoir besoin d'étudier ces concepts.

La classe UIObject implémente les éléments suivants :

- Styles
- Evénements
- Redimensionnement par mise à l'échelle

Pour utiliser les méthodes et propriétés de la classe UIObject, appelez-les directement du composant que vous utilisez. Par exemple, pour appeler la méthode `UIObject.setSize()` 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 l'architecture v2 des composants Macromedia 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`	Position du bord inférieur de l'objet par rapport au bord inférieur de son parent. Lecture seule.
`UIObject.height`	La hauteur de l'objet, en pixels. Lecture seule.
`UIObject.left`	La position gauche de l'objet, en pixels. Lecture seule.
`UIObject.right`	La position du bord droit de l'objet par rapport au bord droit de son parent. Lecture seule.
`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. Lecture seule.
`UIObject.visible`	Valeur booléenne indiquant si l'objet est visible (`true`) ou non (`false`).
`UIObject.width`	La largeur de l'objet, en pixels. Lecture seule.
`UIObject.x`	La position gauche de l'objet, en pixels. Lecture seule.
`UIObject.y`	Renvoie la position du bord supérieur de l'objet par rapport à son parent. Lecture seule.

Evénements de la classe UIObject

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

UIObject.bottom

Disponibilité

Flash Player 6 version 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. Pour définir cette propriété, appelez la méthode UIObject.move().

Exemple

Cet exemple permet de déplacer la case à cocher pour qu'elle soit alignée sous le bord inférieur de la zone de liste :

```
maCaseAcocher.move(maCaseAcocher.x, form.height - listbox.bottom);
```

UIObject.createObject()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

```
occurrenceDeComposant.createObject(nomLiaison, nomOccurrence, profondeur,
  objetInit)
```

Paramètres

nomLiaison Une chaîne indiquant l'identificateur de liaison d'un symbole dans le panneau Bibliothèque.

nomOccurrence 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 version 79.

Edition

Flash MX 2004.

Usage

```
occurrenceDeComposant.createClassObject(nomClasse, nomOccurrence, profondeur,
  objetInit)
```

Paramètres

nomClasse Un objet indiquant la classe de la nouvelle occurrence.

nomOccurrence 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. Effectuez l'une des opérations 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 version 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 version 79.

Edition

Flash MX 2004.

Usage

```
on(draw){
  ...
}
objetDécoute = new Object();
objetDécoute.draw = function(objetEvt){
  ...
}
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 (dans ce cas, draw) qui est géré par une fonction (appelée aussi *gestionnaire*) sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

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);
```

Voir aussi

`UIEventDispatcher.addEventListener()`

UIObject.height

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

`occurrenceDeComposant``.height`

Description

Propriété (lecture seule) : un nombre indiquant la hauteur de l'objet, en pixels. Pour modifier la propriété `height`, appelez la propriété `UIObject.setSize()`.

Exemple

L'exemple suivant rend la case à cocher plus grande :

```
maCaseAcocher.setSize(maCaseAcocher.width, maCaseAcocher.height + 10);
```

UIObject.hide

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

```
on(hide){
  ...
}
objetDécoute = new Object();
objetDécoute.hide = function(objetEvt){
  ...
}
occurrenceDeComposant.addEventListener("hide", objetDécoute)
```

Description

Evénement : diffusé lorsque la propriété `visible` de l'objet passe de `true` à `false`.

Exemple

Le gestionnaire suivant affiche un message dans le panneau de sortie lorsque l'objet auquel il est associé devient invisible.

```
on(hide) {
  trace("Je suis devenu invisible.");
}
```

Voir aussi

`UIObject.reveal`

UIObject.getStyle()

Disponibilité

Flash Player 6 version 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 version 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 version 79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant`.left`

Description

Propriété (lecture seule) : nombre indiquant la longueur du bord gauche de l'objet en pixels par rapport à son parent. Pour définir cette propriété, appelez la méthode `UIObject.move()`.

UIObject.load

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(load){
   ...
}
```

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.load = function(objetEvt){
   ...
}
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 (dans ce cas, load) qui est géré par une fonction (appelée aussi *gestionnaire*) sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

Exemple

L'exemple suivant crée une occurrence de MonSymbole une fois que l'occurrence `form` est chargée :

```
formListener.handleEvent = function(objEvt)
{
  form.createObject("MonSymbole", "sym1", 0);
}
form.addEventListener("load", formListener);
```

UIObject.move

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(move){
  ...
}
```

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.move = function(objetEvt){
  ...
}
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 (dans ce cas, move) qui est géré par une fonction (appelée aussi *gestionnaire*) sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

Exemple

L'exemple suivant appelle la méthode `move()` pour conserver `form2` à 100 pixels vers le bas, à droite de `form1` :

```
formListener.handleEvent = function(){
  form2.move(form1.x + 100, form1.y + 100);
}
form1.addEventListener("move", formListener);
```

UIObject.move()

Disponibilité

Flash Player 6 version 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.

Exemple

Cet exemple permet de déplacer la case à cocher de 10 pixels vers la droite :

```
maCaseAcocher.move(maCaseAcocher.x + 10, maCaseAcocher.y);
```

UIObject.redraw()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.redraw(*always*)

Paramètres

always Valeur Booléenne. Si `true`, dessine l'objet même si `invalidate()` n'a pas été appelé. Si `false`, dessine l'objet uniquement si `invalidate()` a été appelé.

Renvoie

Rien.

Description

Méthode : force la validation de l'objet pour qu'il soit dessiné dans l'image actuelle.

Exemple

L'exemple suivant crée une case à cocher et un bouton et les dessine car il n'est pas prévu que d'autres scripts modifient le formulaire :

```
form.createClassObject(mx.controls.CheckBox, "cb", 0);
form.createClassObject(mx.controls.Button, "b", 1);
form.redraw(true)
```

UIObject.resize

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(resize){
  ...
}
```

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.resize = function(objetEvt){
  ...
}
occurrenceDeComposant.addEventListener("resize", objetDécoute)
```

Description

Evénement : signale aux écouteurs que l'objet a été redimensionné.

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 (dans ce cas, resize) qui est géré par une fonction (appelée aussi *gestionnaire*) sur 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 transmet automatiquement un objet événement (*objetEvt*) 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 événement, consultez *Objets événement*, page 592.

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(objEvt)
  form.sym1.setSize(sym1.width / 2, sym1.height / 4);
}
form.addEventListener("resize", formListener);
```

UIObject.reveal

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

```
on(reveal){
  ...
}
objetDécoute = new Object();
objetDécoute.reveal = function(objetEvt){
  ...
}
occurrenceDeComposant.addEventListener("reveal", objetDécoute)
```

Description

Evénement : diffusé lorsque la propriété `visible` de l'objet passe de `false` à `true`.

Exemple

Le gestionnaire suivant affiche un message dans le panneau de sortie lorsque l'objet auquel il est associé devient visible.

```
on(reveal) {
  trace("Je suis devenu visible.");
}
```

Voir aussi

`UIObject.hide`

UIObject.right

Disponibilité

Flash Player 6 version 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. Pour définir cette propriété, appelez la méthode `UIObject.move()`.

Exemple

L'exemple suivant permet de déplacer la case à cocher afin qu'elle s'aligne sous le bord droit de la zone de liste :

```
maCaseAcocher.move(form.width - listbox.right, maCaseAcocher.y);
```

UIObject.scaleX

Disponibilité

Flash Player 6 version 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 version 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 version 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 version 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 version 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 version 79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.top

Description

Propriété (lecture seule) : nombre indiquant le bord supérieur de l'objet en pixels par rapport à son parent. Pour définir cette propriété, appelez la méthode `UIObject.move()`.

UIObject.unload

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(unload){
   ...
}
```

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.unload = function(objetEvt){
  ...
}
occurrenceDeComposant.addEventListener("unload", objetDécoute)
```

Description

Evénement : indique aux écouteurs que les sous-objets de cet objet sont purgés.

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 (dans ce cas, unload) qui est géré par une fonction (appelée aussi *gestionnaire*) sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

Exemple

L'exemple suivant supprime `sym1` lorsque l'événement `unload` est déclenché :

```
formListener.handleEvent = function(objEvt)
  // objEvt.target est le composant ayant généré l'événement change,
  form.destroyObject(sym1);
}
form.addEventListener("unload", formListener);
```

UIObject.visible

Disponibilité

Flash Player 6 version 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 version 79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.width

Description

Propriété (lecture seule) : un nombre indiquant la largeur de l'objet, en pixels. Pour modifier la largeur, appelez la méthode `UIObject.setSize()`.

Exemple

L'exemple suivant rend la case à cocher plus large :

```
maCaseAcocher.setSize(maCaseAcocher.width + 10, maCaseAcocher.height);
```

UIObject.x

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.x

Description

Propriété (lecture seule) : un nombre indiquant le bord gauche de l'objet, en pixels. Pour définir cette propriété, appelez la méthode `UIObject.move()`.

Exemple

L'exemple suivant permet de déplacer la case à cocher de 10 pixels vers la droite :

```
maCaseAcocher.move(maCaseAcocher.x + 10, maCaseAcocher.y);
```

UIObject.y

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

occurrenceDeComposant.y

Description

Propriété (lecture-seule) ; un nombre indiquant le bord supérieur de l'objet, en pixels. Pour définir cette propriété, appelez la méthode `UIObject.move()`.

Exemple

L'exemple suivant permet de déplacer la case à cocher de 10 pixels vers le bas :

```
maCaseAcocher.move(maCaseAcocher.x, maCaseAcocher.y + 10);
```

Classes de service Web (Flash Professionnel uniquement)

Les classes qui se trouvent dans le paquet mx.services sont des classes d'accès aux services web utilisant le protocole standard SOAP (Simple Object Access Protocol). Cet API WebService n'est pas le même que l'API WebServiceConnector. Le premier est un ensemble de classes que vous pouvez utiliser uniquement dans le code ActionScript et il est commun à plusieurs produits Macromedia. Le dernier est un API appartenant uniquement à Flash MX 2004 : il fournit une interface ActionScript à l'outil de programmation visuelle pour le composant WebServiceConnector.

Rendre les classes du service web disponibles lors de l'exécution (Flash Professionnel uniquement)

Afin de rendre les classes de service web disponibles lors de l'exécution, le composant WebServiceClasses doit se trouver dans la bibliothèque de votre fichier FLA. Ce composant contient les classes d'exécution vous permettant de travailler avec les services web. Pour obtenir des informations détaillées sur l'ajout de ces classes dans votre fichier FLA, consultez « Utilisation de la liaison des données et des services web à l'exécution (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Remarque : Ces classes deviennent automatiquement disponibles lorsque vous ajoutez un composant WebServiceConnector dans votre fichier FLA.

Classes du paquet mx.services (Flash Professionnel uniquement)

Le tableau suivant répertorie les classes du paquet mx.services. Ces classes sont étroitement intégrées. Si vous découvrez ce paquet pour la première fois, vous pouvez en savoir plus en suivant l'ordre de ce tableau.

Classe	Description
Classe WebService (Flash Professionnel uniquement)	Utilisation d'un fichier WSDL qui définit le service web, construit un nouvel objet WebService pour appeler des méthodes de service web et gérer des rappels à partir du service web.
Classe PendingCall (Flash Professionnel uniquement)	Objet renvoyé à la suite d'un appel de méthode de service web que vous implémentez pour gérer les résultats et les erreurs de l'appel.
Classe Log (Flash Professionnel uniquement)	Objet facultatif utilisé pour enregistrer l'activité associée à un objet WebService.
Classe SOAPCall (Flash Professionnel uniquement)	Classe améliorée qui contient des informations sur le fonctionnement du service web et permet de contrôler certains comportements.

Classe Log (Flash Professionnel uniquement)

La classe Log fait partie du paquet mx.services et doit être utilisée avec la classe WebService (consultez *Classe WebService (Flash Professionnel uniquement)*, page 627). Pour obtenir une vue d'ensemble des classes du paquet mx.data.services, consultez *Classes de service Web (Flash Professionnel uniquement)*, page 611.

Vous pouvez créer un nouvel objet Log pour enregistrer l'activité associée à un objet WebService. Pour exécuter le code lors de l'envoi des messages vers un objet Log, utilisez la fonction de rappel `Log.onLog()`. Il n'existe aucun fichier journal ; le mécanisme de consignation est celui utilisé dans le rappel `onLog()` (envoi des messages de consignation vers une commande de trace, par exemple).

Le constructeur crée un objet Log pouvant être transmis comme argument facultatif au constructeur WebService (consultez *Classe WebService (Flash Professionnel uniquement)*, page 627).

Nom de classe ActionScript mx.services.Log

Rappels de l'objet Log

Rappel	Description
`Log.onLog()`	Envoie un message de consignation vers un objet Log.

Constructeur de la classe Log

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

monLogSrvceWeb = new Log([niveauJournal] [, nomJournal]);

Paramètres

niveauJournal Niveau de journal indiquant les types d'informations que vous souhaitez enregistrer dans le journal. Dans le code des services web, les messages de consignation sont divisés en catégories ou niveaux. Le paramètre `niveauJournal` du constructeur de l'objet Log fait référence à ces catégories. Trois `niveaux` sont disponibles :

- `Log.BRIEF` : Le journal enregistre les notifications d'erreur et d'événement de cycle de vie primaire.
- `Log.VERBOSE` : Le journal enregistre toutes les notifications d'erreur et d'événement de cycle de vie.
- `Log.DEBUG` : Le journal enregistre les événements et erreurs de mesure et de granularité fine.

Le `niveau de journal` par défaut est `log.BRIEF`.

nomJournal Nom facultatif inclus dans chaque message de consignation. Si vous utilisez plusieurs objets Log, vous pouvez utiliser cet élément pour identifier le journal ayant enregistré un message donné.

Renvoie

Rien.

Description

Constructeur : crée un objet Log. Utilisez ce constructeur pour créer un journal. Une fois l'objet log créé, vous pouvez le transmettre à un service web pour recevoir des messages.

Exemple

Vous pouvez appeler le constructeur `new Log`, qui renvoie un objet Log à transmettre à votre service web :

```
// crée un nouvel objet Log
monLogSrvceWeb = new Log();
monLogSrvceWeb.onLog = function(txt)
{
  maTrace(txt)
}
```

Vous transmettez ensuite cet objet Log en tant que paramètre au constructeur WebService :

```
monSrvceWeb = new WebService("http://www.myco.com/info.wsdl", monLogSrvceWeb);
```

A chaque exécution du code des services web, des messages sont envoyés vers l'objet Log et la fonction `onLog()` de ce dernier est appelée. Il s'agit du seul endroit possible pour placer le code qui affiche les messages de consignation si vous voulez les voir en temps réel.

Voici des exemples de messages de consignation :

```
7/30 15:22:43 [INFO] SOAP: Decoding PendingCall response
7/30 15:22:43 [DEBUG] SOAP: Decoding SOAP response envelope
7/30 15:22:43 [DEBUG] SOAP: Decoding SOAP response body
7/30 15:22:44 [INFO] SOAP: Decoded SOAP response into result [16 millis]
7/30 15:22:46 [INFO] SOAP: Received SOAP response from network [6469 millis]
7/30 15:22:46 [INFO] SOAP: Parsed SOAP response XML [15 millis]
7/30 15:22:46 [INFO] SOAP: Decoding PendingCall response
7/30 15:22:46 [DEBUG] SOAP: Decoding SOAP response envelope
7/30 15:22:46 [DEBUG] SOAP: Decoding SOAP response body
7/30 15:22:46 [INFO] SOAP: Decoded SOAP response into result [16 millis]
```

Log.onLog()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monLogSrvceWeb.onLog = function(message)
```

Paramètres

`message` Le message de consignation transmis au gestionnaire. Pour plus d'informations sur les messages de consignation, consultez *Classe Log (Flash Professionnel uniquement)*, page 612.

Renvoie

Aucun.

Description

Fonction de rappel du journal : Flash Player appelle cette fonction lorsqu'un message de consignation est envoyé vers un fichier journal. Ce gestionnaire est l'endroit idéal pour placer le code d'enregistrement ou d'affichage des messages de consignation (commandes `trace`, par exemple). Pour obtenir une description du processus de construction d'un journal, consultez *Classe Log (Flash Professionnel uniquement)*, page 612.

Exemple

L'exemple suivant permet de créer un nouveau journal, de le transmettre à un nouvel objet WebService et de gérer les messages de consignation.

```
// crée un nouvel objet Log
monLogSrvceWeb = new Log();
// transmet le journal au service web
monServiceWeb = new WebService(URIwsdl, monLogSrvceWeb);
// gère les messages de consignation entrants
monLogSrvceWeb.onLog = function(message)
{
  matrace("Log Event:\r monLogSrvceWeb.message="+message+);
}
```

Classe PendingCall (Flash Professionnel uniquement)

La classe PendingCall fait partie du paquet mx.services et doit être utilisée avec la classe WebService (consultez *Classe WebService (Flash Professionnel uniquement)*, page 627). Pour obtenir une vue d'ensemble des classes du paquet mx.data.services, consultez *Classes de service Web (Flash Professionnel uniquement)*, page 611.

Lorsque vous appelez une méthode sur un objet WebService, celui-ci renvoie un objet PendingCall. L'objet PendingCall n'est pas construit par le développeur. Les rappels `onResult` et `onFault` de l'objet PendingCall permettent de gérer la réponse asynchrone de la méthode du service web. Si la méthode du service web renvoie une erreur, Flash Player appelle le rappel `PendingCall.onFault` et transmet un objet SOAPFault représentant l'erreur XML SOAP renvoyée par le serveur/service web. Si l'invocation du service web est réussie, Flash Player appelle le rappel `PendingCall.onResult` et transmet un objet résultat. L'objet résultat est la réponse XML du service web décodée ou convertie dans ActionScript. Pour plus d'informations sur l'objet WebService, consultez *Classe WebService (Flash Professionnel uniquement)*, page 627.

L'objet PendingCall vous permet également d'accéder aux paramètres de sortie lorsqu'il en existe plusieurs. Si la plupart des services web ne renvoient qu'un seul résultat, certains en renvoient plusieurs. La valeur renvoyée désignée dans cet API correspond simplement au premier (ou unique) résultat. Les fonctions `PendingCall.getOuptutXXX` vous permettent d'accéder à l'ensemble des résultats et non pas seulement au premier. Aussi, tandis que la valeur renvoyée vous est transmise dans l'argument au rappel `onResult()`, vous devez utiliser les méthodes `getOutputValues()` (renvoie un tableau) et `getOutputValue(index)` (en renvoie un individuel) pour obtenir les valeurs décodées ActionScript.

Vous pouvez aussi accéder directement à l'objet SOAPParameter. L'objet SOAPParameter est un objet ActionScript doté de deux propriétés : la propriété `value` contient la valeur ActionScript d'un paramètre de sortie, tandis que la propriété `element` contient la valeur XML du paramètre de sortie. Les fonctions suivantes renvoient un objet SOAPParameter ou un tableau d'objets SOAPParameter contenant la valeur (`param.value`) ainsi que l'élément XML (`param.element`) : `getOutputParameters()`, `getOutputParameterByName(nom)` et `getOutputParameter(index)`.

Nom de classe ActionScript mx.services.PendingCall

Fonctions de l'objet PendingCall

Fonction	Description
`PendingCall.getOutputParameter()`	Obtient un objet SOAPParameter en fonction de l'`index` transmis.
`PendingCall.getOutputParameterByName()`	Obtient un objet SOAPParameter en fonction du `nomLocal` transmis.
`PendingCall.getOutputParameters()`	Obtient un tableau des objets SOAPParameter.
`PendingCall.getOutputValue()`	Obtient la valeur de sortie en fonction de l'index transmis.
`PendingCall.getOutputValues()`	Obtient un tableau de l'ensemble des valeurs de sortie.

Propriétés de l'objet PendingCall

Propriété	Description
`PendingCall.myCall`	Descripteur d'opération SOAPCall pour l'opération PendingCall.
`PendingCall.request`	Requête SOAP au format XML brut.
`PendingCall.response`	Réponse SOAP au format XML brut.

Rappels de l'objet PendingCall

Rappel	Description
`PendingCall.onFault()`	Appelé par un service web en cas d'échec de la méthode.
`PendingCall.onResult()`	Appelé si la méthode réussir et renvoie un résultat.

Constructeur pour la classe PendingCall

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Description

L'objet PendingCall n'est pas construit par le développeur. A la place, lorsque vous appelez une fonction sur un objet `WebService`, l'objet `WebService` renvoie un objet `PendingCall`.

PendingCall.getOutputParameter()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monAppelEnAttente.getOutputParameter(var index)
```

Paramètres

index Index du paramètre.

Renvoie

Un objet SOAPParameter avec les éléments suivants :

Elément	Description
value	Valeur ActionScript du paramètre.
element	Valeur XML brute du paramètre dans l'enveloppe SOAP.

Description

Fonction : obtient un paramètre de sortie supplémentaire de l'objet SOAPParameter, contenant la valeur et l'élément XML. Les appels RPC SOAP peuvent renvoyer plusieurs paramètres de sortie. La première (ou unique) valeur renvoyée vous est toujours transmise dans l'argument results du rappel `onResult()` ; pour accéder aux autres, vous devez utiliser les fonctions telles que celle-ci ou `getOutputValue()`. La fonction `getOutputParameter()` renvoie le paramètre de sortie *nth* en tant qu'objet SOAPParameter.

Consultez également `PendingCall.getOutputValue()`, `PendingCall.getOutputValues()`, `PendingCall.getOutputParameterByName()` et `PendingCall.getOutputParameters()`.

Exemple

En considérant le fichier descripteur SOAP ci-dessous, `getOutputParameter(1)` renverrait un objet SOAPParameter avec les paramètres `value="Bonjour !"` et `element=the <outParam2> XMLNode`.

```
...
<SOAP:Body>
  <rpcResponse>
    <outParam1 xsi:type="xsd:int">54</outParam1>
    <outParam2 xsi:type="xsd:string">Bonjour !</outParam2>
    <outParam3 xsi:type="xsd:boolean">true</outParam3>
  </rpcResponse>
</SOAP:Body>
...
```

PendingCall.getOutputParameterByName()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monAppelEnAttente.getOutputParameterByName(var nomLocal)
```

Paramètres

nomLocal Nom local du paramètre. En d'autres termes, le nom d'un élément XML, dépouillé de toute information sur l'espace de nom. Par exemple, le nom local des deux éléments suivants est `bob` :

```
<bob abc="123">
<xsd:bob def="ghi">
```

Renvoie

Un objet SOAPParameter avec les éléments suivants :

Elément	Description
value	Valeur ActionScript du paramètre.
element	Valeur XML brute du paramètre dans l'enveloppe SOAP.

Description

Fonction : obtient n'importe quel paramètre de sortie en tant qu'objet SOAPParameter, contenant la valeur et l'élément XML. Les appels RPC SOAP peuvent renvoyer plusieurs paramètres de sortie. La première (ou unique) valeur renvoyée vous est toujours transmise dans l'argument results du rappel `onResult()` ; pour accéder aux autres, vous devez utiliser les API telles que celle-ci. L'appel `getOutputParameterByName()` renvoie le paramètre de sortie avec le nom *nomLocal*.

Consultez également `PendingCall.getOutputValue()`, `PendingCall.getOutputValues()`, `PendingCall.getOutputParameter()` et `PendingCall.getOutputParameters()`.

Exemple

En considérant le fichier descripteur SOAP ci-dessous, `getOutputParameterByName("outParam2")` renverrait un objet SOAPParameter avec les paramètres `value="Bonjour!"` et `element=the <outParam2> XMLNode`.

```
...
<SOAP:Body>
  <rpcResponse>
    <outParam1 xsi:type="xsd:int">54</outParam1>
    <outParam2 xsi:type="xsd:string">Bonjour !</outParam2>
    <outParam3 xsi:type="xsd:boolean">true</outParam3>
  </rpcResponse>
</SOAP:Body>
...
```

PendingCall.getOutputParameters()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monAppelEnAttente.getOutputParameterByName()
```

Paramètres

Aucun.

Renvoie

Tableau des objets SOAPParameter avec les éléments suivants :

Elément	Description
value	Valeur ActionScript du paramètre.
element	Valeur XML brute du paramètre dans l'enveloppe SOAP.

Description

Fonction : obtient des paramètres de sortie supplémentaires de l'objet SOAPParameter, contenant les valeurs et les éléments XML. Les appels RPC SOAP peuvent renvoyer plusieurs paramètres de sortie. La première (ou unique) valeur renvoyée vous est toujours transmise dans l'argument results du rappel `onResult()` ; pour accéder aux autres, vous devez utiliser les API telles que celle-ci ou `getOutputValues()`.

Consultez également `PendingCall.getOutputValue()`, `PendingCall.getOutputValues()`, `PendingCall.getOutputParameterByName()` et `PendingCall.getOutputParameter()`.

PendingCall.getOutputValue()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monAppelEnAttente.getOutputValue(var index)
```

Paramètres

index Index d'un paramètre de sortie. Le premier paramètre est l'index `0`.

Renvoie

Le paramètre de sortie *nth*.

Description

Fonction : obtient la valeur ActionScript décodée d'un paramètre de sortie individuel. Les appels RPC SOAP peuvent renvoyer plusieurs paramètres de sortie. La première (ou unique) valeur renvoyée vous est toujours transmise dans l'argument results du rappel `onResult()` ; pour accéder aux autres, vous devez utiliser les API telles que celles-ci ou `getOutputParameter()`. L'appel `getOutputValue()` renvoie le paramètre de sortie *`nth`*.

Consultez également `PendingCall.getOutputParameter()`, `PendingCall.getOutputValues()`, `PendingCall.getOutputParameterByName()` et `PendingCall.getOutputParameters()`.

Exemple

En considérant le fichier descripteur SOAP ci-dessous, `getOutputValue(2)` renverrait la valeur `true`.

```
...
<SOAP:Body>
  <rpcResponse>
    <outParam1 xsi:type="xsd:int">54</outParam1>
    <outParam2 xsi:type="xsd:string">Bonjour !</outParam2>
    <outParam3 xsi:type="xsd:boolean">true</outParam3>
  </rpcResponse>
</SOAP:Body>
...
```

PendingCall.getOutputValues()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

`monAppelEnAttente``.getOutputValues()`

Paramètres

Aucun.

Renvoie

Tableau regroupant les valeurs décodées de tous les paramètres de sortie.

Description

Fonction : obtient la valeur décodée ActionScript de tous les paramètres de sortie. Les appels RPC SOAP peuvent renvoyer plusieurs paramètres de sortie. La première (ou unique) valeur renvoyée vous est toujours transmise dans l'argument results du rappel `onResult()` ; pour accéder aux autres, vous devez utiliser les API telles que celle-ci ou `getOutputParameters()`.

Consultez également `PendingCall.getOutputValue()`, `PendingCall.getOutputParameter()`, `PendingCall.getOutputParameterByName()` et `PendingCall.getOutputParameters()`.

PendingCall.myCall

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

appelEnAttente.myCall

Description

Propriété : l'objet SOAPCall correspondant à l'opération PendingCall. L'objet SOAPCall contient des informations sur le fonctionnement du service web et permet de contrôler certains comportements. Pour plus d'informations, consultez *Classe SOAPCall (Flash Professionnel uniquement)*, page 624.

Exemple

Le rappel `onResult` suivant recherche le nom de l'opération SOAPCall.

```
rappel.onResult = function(résultat)
{
  // Vérifier le nom de mon opération
  trace("Le nom de mon opération est " + this.myCall.name);
}
```

PendingCall.onFault()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
objMonAppelEnAttente.onFault = function(erreur)
{
  // gère toutes les erreurs, par exemple en indiquant à
  // l'utilisateur que le serveur n'est pas disponible ou en lui conseillant
  de contacter le service
  // d'assistance technique

}
```

Paramètres

erreur Version de l'objet ActionScript décodé de l'erreur avec les propriétés. Si les informations relatives à l'erreur proviennent d'un serveur sous la forme de fichier XML, alors l'objet SOAPFault correspond à la version ActionScript décodée de cet XML.

Le type d'objet erreur renvoyé à `PendingCall.onfault()` est un objet SOAPFault. Il n'est pas directement construit par des développeurs, mais renvoyé suite à un échec. Il s'agit de la correspondance dans ActionScript du type XML SOAP Fault.

SOAPFault, propriété	Description
faultcode	Chaîne : chaîne courte décrivant l'erreur.
faultstring	Chaîne : description de l'erreur sous forme lisible par un humain.
detail	Chaîne : informations spécifiques à l'application associée à l'erreur, par exemple trace de pile ou autres informations renvoyées par le moteur du service web.
element	XML : l'objet XML représentant la version XML de l'erreur.
faultactor	Chaîne : source de l'erreur (facultatif si aucun intermédiaire n'est impliqué).

Renvoie

Rien.

Description

Fonction de rappel de l'objet PendingCall ; vous fournissez ce gestionnaire que Flash Player appelle lorsqu'une méthode du service web a échoué et renvoyé une erreur. Le paramètre par défaut est un objet SOAPFault ActionScript.

Exemple

L'exemple suivant permet de gérer des erreurs renvoyées à partir de la méthode du service web.

```
// gérer toute erreur renvoyée par l'utilisation d'une méthode de service web.
objMonAppelEnAttente = monServiceWeb.nomDeLaMéthode(params)
monAppelEnAttenteObj.onFault = function(erreur)
{
  // intercepter l'erreur SOAP
  DebugOutputField.text = fault.faultstring;

  // ajoutez le code pour gérer toutes les erreurs, par exemple, en indiquant
  // à l'utilisateur que le serveur n'est pas disponible ou en lui conseillant
  de contacter le service
  // d'assistance technique

}
```

PendingCall.onResult()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
objMonAppelEnAttente.onResult = function(résultat)
{
  // intercepte le résultat et le gère pour cette application

}
```

Paramètres

résultat Version objet ActionScript décodée du résultat XML renvoyé par une méthode de service web appelée avec `objMonAppelEnAttente = monServiceWeb.`*`nomDeLaMéthode`*`(`*`params`*`)`.

Renvoie

Aucun.

Description

Fonction de rappel de PendingCall ; vous fournissez ce gestionnaire que Flash Player appelle lorsqu'une méthode de service web réussit et renvoie un résultat. Le résultat est une version objet ActionScript décodée du XML renvoyé par l'opération. Pour obtenir le renvoi du résultat XML au format brut au lieu du format décodé, accédez à la propriété *PendingCall*.`response` (consultez *PendingCall.response*).

Exemple

L'exemple suivant permet de gérer les résultats renvoyés à partir de la méthode du service web.

```
// gère les résultats renvoyés par l'utilisation d'une méthode de service web
objMonAppelEnAttente = monServiceWeb.nomDeLaMéthode(params)
objMonAppelEnAttente.onResult = function(résultat)
{
  // intercepter le résultat et le gérer pour cette application
  ResultOutputField.text = result;
}
```

PendingCall.request

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
rawXML = monRappelEnAttente.request;
```

Description

Propriété PendingCall : contient le formulaire XML brut de la requête actuelle envoyée avec `monRappelEnAttente = monServiceWeb.nomDeLaRequête()`. Normalement, vous ne devriez pas avoir besoin d'utiliser la propriété `PendingCall.request`, mais vous le pouvez si vous souhaitez accéder aux données SOAP envoyées. Utilisez-la pour accéder au XML brut de la demande. Utilisez `monRappelEnAttente.onResult()` pour obtenir la version ActionScript des résultats de la demande.

PendingCall.response

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
rawXML = monRappelEnAttente.response;
```

Description

Propriété PendingCall : contient le formulaire XML brut de la réponse à l'appel de la méthode de service web le plus récent envoyé avec `monRappelEnAttente = monServiceWeb.nomDeLaMéthode()`. Normalement, vous ne devriez pas avoir besoin d'utiliser la propriété `PendingCall.response`, mais vous le pouvez si vous souhaitez accéder aux données SOAP envoyées. Utilisez `monRappelEnAttente.onResult()` pour obtenir la version ActionScript des résultats de la demande.

Classe SOAPCall (Flash Professionnel uniquement)

La classe SOAPCall fait partie du paquet mx.services et est conçue comme une fonction améliorée à utiliser avec la classe WebService (consultez *Classe WebService (Flash Professionnel uniquement)*, page 627). Pour obtenir une vue d'ensemble des classes du paquet mx.data.services, consultez *Classes de service Web (Flash Professionnel uniquement)*, page 611.

Lorsque vous créez un nouvel objet WebService, il contient les méthodes correspondant aux opérations de l'URL du fichier WSDL que vous transmettez. Derrière les séquences, un objet SOAPCall est également créé pour chaque opération dans le fichier WSDL. SOAPCall est le descripteur de l'opération et en tant que tel, il contient toutes les informations sur cette opération spécifique (comment XML doit apparaître sur la connexion, le style d'opération, etc.). Il permet aussi de contrôler certains comportements. Vous pouvez obtenir SOAPCall pour une opération particulière en utilisant la fonction `getCall(nomOpération)`. lI existe un SOAPCall unique pour chaque opération, partagé par l'ensemble des appels actifs de cette opération. Dès que vous obtenez SOAPCall, vous pouvez personnaliser le descripteur en effectuant les opérations suivantes :

- Activer/Désactiver le décodage de la réponse XML.
- Activer/Désactiver le délai de conversion des tableaux SOAP en objets ActionScript.

- Modifier la configuration simultanée d'une opération spécifiée.
- Ajouter un en-tête à l'objet SOAPCall.

Nom de classe ActionScript mx.services.SOAPCall

Fonctions de l'objet SOAPCall

Fonction	Description
`SOAPCall.addHeader()`	Ajoute un en-tête à l'objet SOAPCall.

Propriétés de l'objet SOAPCall

Propriété	Description
`SOAPCall.concurrency`	Modifie la configuration simultanée d'une opération spécifiée.
`SOAPCall.doDecoding`	Active/Désactive le décodage de la réponse XML.
`SOAPCall.doLazyDecoding`	Active/Désactive le délai de transformation des tableaux SOAP en objets ActionScript.

Constructeur de la classe SOAPCall

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Description

L'objet SOAPCall n'est pas construit par le développeur. Lorsque vous appelez une méthode sur un objet WebService, l'objet WebService renvoie un objet PendingCall. Pour accéder à l'objet SOAPCall associé, utilisez `monAppelEnAttente.myCall`.

SOAPCall.addHeader()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
SOAPCall.addHeader(var en-tête)
```

Paramètres

en-tête En-tête à ajouter.

Renvoie

Aucun.

Description

Fonction : ajoute un en-tête à l'objet SOAPCall.

Exemple

L'exemple suivant crée un nouvel en-tête SOAP et le joint à SOAPCall. Le code suivant :

```
import mx.services.QName;

var qname = new QName("bar", "http://foo");
var value = "Bonjour!";
var en-tête = new SOAPHeader(qname, valeur);
soapCall.addHeader(en-tête);
```

crée l'en-tête SOAP suivant :

```
...
<SOAP:Header>
  <ns1:bar
    xmlns:ns1="http://foo"
    xsi:type="xsd:string">Bonjour!</ns1:bar>
</SOAP:Header>
...
```

SOAPCall.concurrency

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

SOAPCall.concurrency

Description

Propriété : nombre de demandes simultanées. Les valeurs possibles sont répertoriées dans le tableau ci-dessous :

Valeur	Description
SOAPCall.MULTIPLE_CONCURRENCY	Permettre plusieurs appels actifs.
SOAPCall.SINGLE_CONCURRENCY	Permettre un seul appel à la fois en créant une erreur après un appel actif.
SOAPCall.LAST_CONCURRENCY	Permettre un seul appel en annulant les précédents.

SOAPCall.doDecoding

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

SOAPCall.doDecoding

Description

Propriété : active/désactive le décodage de la réponse XML ; par défaut, la réponse XML est convertie (décodée) en objets ActionScript. Si vous souhaitez uniquement la réponse XML, définissez `SOAPCall.doDecoding = false`.

SOAPCall.doLazyDecoding

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

SOAPCall.doLazyDecoding

Description

Propriété : active/désactive le « décodage paresseux » des tableaux. Par défaut, nous utilisons un algorithme de décodage « paresseux » pour retarder la conversion des tableaux SOAP en objets ActionScript jusqu'au dernier moment : les opérations sont ainsi renvoyées beaucoup plus vite lorsque les jeux de données sont importants. Cela signifie que tout tableau récupéré à distance est un objet ArrayProxy. Lorsque vous accédez à un index particulier (`foo[5]`), cet élément est décodé automatiquement si nécessaire. Il est possible de désactiver ce comportement (tous les tableaux sont alors entièrement décodés) en définissant `SOAPCall.doLazyDecoding = false`.

Classe WebService (Flash Professionnel uniquement)

La classe WebService fait partie du paquet mx.services et doit être utilisée avec les classes suivantes :

- *Classe Log (Flash Professionnel uniquement)*
- *Classe PendingCall (Flash Professionnel uniquement)*
- *Classe SOAPCall (Flash Professionnel uniquement)*

Remarque : Cet API WebService n'est pas le même que l'API WebServiceConnector. Le premier est un ensemble de classes que vous pouvez utiliser uniquement dans le code ActionScript et il est commun à plusieurs produits Macromedia. Le dernier est un API appartenant uniquement à Flash MX 2004 : il fournit une interface ActionScript à l'outil de programmation visuelle pour le composant WebServiceConnector.

Pour obtenir une vue d'ensemble des classes du paquet mx.services, consultez *Classes de service Web (Flash Professionnel uniquement)*, page 611.

L'objet WebServices agit comme référence locale à un service web à distance. Lorsque vous créez un nouvel objet WebService, le fichier WSDL de définition du service web est téléchargé, analysé et placé dans l'objet. Vous pouvez alors appeler les méthodes du service web directement sur l'objet WebService et gérer tous les rappels en provenance du service web. Lorsque le traitement du fichier WSDL a réussi et que l'objet WebService est prêt, le rappel `onLoad()` est invoqué. S'il survient un problème au cours du chargement du fichier WSDL, le rappel `onFault()` est invoqué.

Lorsque vous appelez une méthode sur un objet WebService, la valeur renvoyée est un objet de rappel. Le type d'objet du rappel renvoyé à partir de toutes les invocations de méthode du service web est `PendingCall`. Habituellement, ces objets ne sont pas construits par des développeurs. Ils sont générés automatiquement, suite à la commande `webServiceObject.webServiceMethodName()`. Ces objets ne résultent pas de l'appel WebService qui arrive par la suite. L'objet PendingCall représente l'appel en cours. Lorsque l'opération WebService se termine (quelques secondes après le lancement de l'appel de méthode généralement), les différents champs de données PendingCall sont renseignés et le rappel `onResult` ou `onFault` que vous fournissez est appelé. Pour plus d'informations sur l'objet PendingCall, consultez *Classe PendingCall (Flash Professionnel uniquement)*, page 615.

Le Player met en file d'attente tous les appels effectués avant l'analyse du fichier WSDL et tente de les exécuter après l'analyse. En effet, le fichier WSDL contient des informations nécessaires à un codage correct et à l'envoi d'une requête SOAP. Les appels de fonction que vous effectuez après analyse du fichier WSDL n'ont pas besoin d'être placés dans la file d'attente ; ils se produisent immédiatement. Si un appel mis en file d'attente ne correspond pas au nom de l'une des opérations définies dans le fichier WSDL, Flash Player renvoie une erreur vers l'objet de rappel que vous avez reçu lorsque vous avez lancé votre appel.

Nom de classe ActionScript mx.services.WebService

Utilisation de l'API WebServices

L'API WebServices, incluse dans le paquet mx.services, est constituée des classes WebService, Log et PendingCall.

Types pris en charge

La fonction WebService prend en charge un sous-ensemble de types de schéma XML tels qu'ils sont définis dans les tableaux ci-dessous.

Les types complexes et le type tableau à codage SOAP sont également supportés ; ils peuvent se composer d'autres types complexes, tableaux ou types de schéma XML intégrés :

Types numériques simples

Type de Schéma XML	Liaison ActionScript
`decimal`	Number
`integer`	Number
`negativeInteger`	Number
`nonNegativeInteger`	Number
`positiveInteger`	Number

Type de Schéma XML	Liaison ActionScript
long	Number
int	Number
short	Number
byte	Number
unsignedLong	Number
unsignedShort	Number
unsignedInt	Number
unsignedByte	Number
float	Number
double	Number

Types simples de date et d'heure

Type de Schéma XML	Liaison ActionScript
date	Objet Date
datetime	Objet Date
duration	Objet Date
gDay	Objet Date
gMonth	Objet Date
gMonthDay	Objet Date
gYear	Objet Date
gYearMonth	Objet Date
time	Objet Date

Types simples de nom et de chaîne

Type de Schéma XML	Liaison ActionScript
string	Chaîne ActionScript
normalizedString	Chaîne ActionScript
QName	mx.services.Qname object

Type Booléen

Type de Schéma XML	Liaison ActionScript
Booléen	Booléen

Types d'objet

Type de Schéma XML	Liaison ActionScript
`Tout type`	objet XML
`Type complexe`	Objet ActionScript composé de propriétés de tout type pris en charge
`Tableau`	Tableau ActionScript composé de tout objet ou type pris en charge

Eléments d'objet de schéma XML pris en charge

```
schema
     complexType
          complexContent
               restriction
          sequence | simpleContent
               restriction
     element
          complexType | simpleType
```

Sécurité WebService

L'API WebService est conforme au modèle de sécurité de Flash Player.

Authentification et autorisation utilisateur

Les règles d'authentification et d'autorisation de l'API WebService sont les mêmes que celles employées pour toute opération réseau XML de Flash. Le protocole SOAP ne spécifie aucune méthode d'authentification et d'autorisation. Par exemple, lorsque la couche de transport HTTP sous-jacente renvoie une réponse HTTP BASIC dans les en-têtes HTTP, le navigateur répond en présentant une boîte de dialogue destinée à l'utilisateur et en ajoutant la saisie de l'utilisateur dans les en-têtes HTTP des messages suivants. Ce mécanisme existe à un niveau inférieur à celui du protocole SOAP et fait partie intégrante de l'authentification HTTP dans Flash.

Intégrité du message

La sécurité des messages implique le cryptage des messages SOAP au niveau d'une couche conceptuelle située au-dessus des paquets réseau sur lesquels les messages SOAP sont livrés.

Sécurité du transport Le protocole de transport réseau sous-jacent pour les services web SOAP Flash Player est toujours HTTP POST. Par conséquent, tout moyen de sécurité pouvant être appliqué à la couche de transport HTTP Flash (SSL, par exemple) est géré via des appels des services web de Flash. SSL/HTTPS propose la forme la plus courante de sécurité du transport pour la messagerie SOAP et l'utilisation de l'authentification HTTP BASIC, associée à SSL sur la couche de transport, correspond à la méthode de sécurité la plus couramment utilisée par les sites web.

Fonctions de l'objet WebService

Fonction	Description
`WebService.nomDeMaMéthode()`	Invoque une opération spécifique du service web définie par WSDL.
`WebService.getCall()`	Obtient l'objet SOAPCall pour une opération donnée.

Rappels de l'objet WebService

Rappel	Description
`WebService.onLoad()`	Appelé lorsque le service web a réussi le chargement et l'analyse de son fichier WSDL.
`WebService.onFault()`	Appelé lorsqu'une erreur s'est produite au cours de l'analyse WSDL.

Constructeur de la classe WebService

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monObjetWebService = new WebService(URIwsdl [, objetLog]);
```

Paramètres

Les paramètres constructeur sont les suivants :

URIwsdl URI du fichier WSDL du service web.

objetLog Paramètre facultatif indiquant le nom de l'objet Log (journal) pour ce service web (consultez *Classe Log (Flash Professionnel uniquement)*, page 612).

Renvoie

Aucun.

Description

Pour créer un objet WebService, vous devez appeler la méthode `new WebService()` et fournir l'URI de son fichier WSDL. Flash Player renvoie un objet WebService. Le constructeur de l'objet WebService peut accepter un objet Log et une URL proxy :

```
monObjetWebService = new WebService(URIwsdl [, objetLog]);
```

Si vous le souhaitez, vous pouvez utiliser deux rappels pour l'objet WebService. Flash Player appelle la fonction `webServiceObject.onLoad(WSDLDocument)` une fois l'analyse du fichier WSDL terminée et l'objet complet. Si vous voulez exécuter du code, après l'analyse complète du fichier WSDL uniquement, vous pouvez l'insérer à cet emplacement. Vous pouvez, par exemple, placer votre premier appel de méthode de service web dans cette fonction.

Flash Player appelle `webServiceObject.onFault(fault)` lorsqu'une erreur se produit lors de la recherche ou de l'analyse du fichier WSDL. Il s'agit d'un bon emplacement pour insérer du code de débogage ou du code indiquant à l'utilisateur que le serveur n'est pas disponible, qu'il doit retenter l'opération plus tard ou toute autre information de ce type. Pour plus d'informations, consultez les entrées individuelles correspondant à ces fonctions.

Invocation d'une opération du service web : vous invoquez une opération du service web en tant que méthode directement disponible sur le service web. Par exemple, si votre service web possède la méthode `obtenirInfosSociété(symboleTéléscripteur)`, invoquez alors la méthode de la manière suivante :

```
monObjetAppelEnAttente =
  monObjetWebService.obtenirInfosSociété(symboleTéléscripteur);
```

Dans l'exemple précédent, l'objet de rappel s'appelle `monObjetAppelEnAttente`. Toutes les invocations de méthode sont asynchrones et renvoient un objet de rappel de type `PendingCall`. Le terme « asynchrone » signifie que les résultats de l'appel du service web ne sont pas immédiatement disponibles.

Lorsque vous effectuez l'appel

```
x = serviceBoursier.obtenirLaCotation("macr");
```

l'objet `x` n'est pas le résultat de `obtenirLaCotation` (c'est un objet PendingCall). Les résultats réels ne sont disponibles que plus tard (en général quelques secondes plus tard), lorsque l'opération du service web est terminée. Votre code ActionScript est signalé par un appel à la fonction de rappel `onResult`.

Gestion de l'objet PendingCall : cet objet de rappel est un objet PendingCall qui vous sert à gérer les résultats et les erreurs à partir de la méthode du service web appelée (consultez *Classe PendingCall (Flash Professionnel uniquement)*, page 615). Exemple :

```
monObjetAppelEnAttente = monObjetWebService.nomDeMaMéthode(param1, ...,
  paramN);
monObjetAppelEnAttente.onResult = function(résultat)
{
  champRésultat.text = résultat
}
monObjetAppelEnAttente.onFault = function(erreur)
{
  champDébogage.text = erreur.faultCode + "," + erreur.faultstring;

  // ajoutez le code pour gérer toutes les erreurs, par exemple, en indiquant
  // à l'utilisateur que le serveur n'est pas disponible ou en lui conseillant
  de contacter le service
  // d'assistance technique

}
```

WebService.getCall()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
getCall(var nomOpération)
```

Paramètres

`nomOpération` Opération du service web de l'objet SOAPCall correspondant que vous souhaitez récupérer.

Renvoie

L'objet SOAPCall.

Description

Lorsque vous créez un nouvel objet `WebService`, il contient les méthodes correspondant aux opérations de l'URL du fichier WSDL que vous transmettez. Derrière les séquences, un objet SOAPCall est également créé pour chaque opération dans le fichier WSDL. SOAPCall est le descripteur de l'opération et en tant que tel, il contient toutes les informations sur cette opération spécifique (comment XML doit apparaître sur la connexion, le style d'opération, etc.). Il permet aussi de contrôler certains comportements. Vous pouvez obtenir l'objet SOAPCall pour une opération donnée en utilisant la méthode `getCall(nomOpération)`. lI existe un SOAPCall unique pour chaque opération, partagé par l'ensemble des appels actifs de cette opération. Dès que vous obtenez l'objet SOAPCall, vous pouvez modifier le descripteur d'opérateur à l'aide de l'API SOAPCall. Pour plus d'informations, consultez *Classe SOAPCall (Flash Professionnel uniquement)*, page 624.

Exemple

Pour obtenir un exemple d'utilisation de cet appel, consultez *Classe SOAPCall (Flash Professionnel uniquement)*, page 624.

WebService.onFault()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

`monObjetWebService``.onFault`

Paramètres

`erreur` Version de l'objet ActionScript décodé de l'erreur avec les propriétés. Si les informations relatives à l'erreur proviennent d'un serveur sous la forme de fichier XML, alors l'objet SOAPFault correspond à la version ActionScript décodée de cet XML.

Le type de l'objet erreur renvoyé aux méthodes `webservice.onFault()` est un objet `SOAPFault`. Il n'est pas directement construit par des développeurs, mais renvoyé suite à un échec. Il s'agit de la correspondance dans ActionScript du type XML SOAP Fault.

SOAPFault, propriété	Description
faultcode	Chaîne : QName court standard décrivant l'erreur.
faultstring	Chaîne : description de l'erreur sous forme lisible par un humain.
detail	Chaîne : informations spécifiques à l'application associée à l'erreur, par exemple trace de pile ou autres informations renvoyées par le moteur du service web.

SOAPFault, propriété	Description
element	XML : l'objet XML représentant la version XML de l'erreur.
faultactor	Chaîne : source de l'erreur ; facultative si aucun intermédiaire n'est impliqué.

Renvoie

Rien.

Description

Fonction de rappel WebService : Flash Player appelle cette fonction lorsque la nouvelle méthode `webService(WSDLUrl)` a échoué et a renvoyé une erreur. Cela peut se produire lorsque le fichier WSDL ne peut pas être analysé ou lorsqu'il est introuvable. Le paramètre par défaut est un objet ActionScript SOAPFault.

Exemple

L'exemple suivant permet de gérer toute erreur renvoyée suite à la création de l'objet WebService.

```
monObjetWebService.onFault = function(erreur)
{
  // capture l'erreur
  DebugOutputField.text = fault.faultstring;

  // ajoutez le code pour gérer toutes les erreurs, par exemple, en indiquant
  // à l'utilisateur que le serveur n'est pas disponible ou en lui conseillant
  de contacter le service
  // d'assistance technique

}
```

WebService.onLoad()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
monService.onLoad
```

Paramètres

Documentwsdl document XML WSDL.

Renvoie

Aucun.

Description

Fonction de rappel Webservice : Flash Player utilise ce rappel lorsque l'objet WebService a réussi le chargement et l'analyse de son fichier WSDL. Les opérations peuvent être invoquées dans une application avant que cet événement ne survienne ; lorsqu'il se produit, elles sont mises en file d'attente en interne et ne sont pas transmises tant que le chargement du fichier WSDL n'a pas eu lieu.

Exemple

L'exemple suivant spécifie l'URI du fichier WSDL, crée un nouvel objet WebService et reçoit le document WSDL après chargement.

```
// préciser l'URI du fichier WSDL
var URIwsdl = "http://www.flash-db.com/services/ws/companyInfo.wsdl";

// crée un nouvel objet de service web
serviceBoursier = new WebService(URIwsdl);

// reçoit le document WSDL après chargement
serviceBoursier.onLoad = function(Documentwsdl);
{
  // code à exécuter lorsque le chargement du fichier WSDL est terminé et
  lorsque
  // l'objet a été créé
}
```

WebService.*nomDeMaMéthode*()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
objetDeRappel = monObjetWebService.nomDeMaMéthode(param1, ... paramN);
```

Paramètres

Les paramètres requis dépendent de la méthode de service web appelée.

Renvoie

objetDeRappel Objet PendingCall auquel vous pouvez associer une fonction pour gérer les résultats et les erreurs lors de l'invocation. Pour plus d'informations, consultez *Classe PendingCall (Flash Professionnel uniquement)*, page 615.

Le rappel invoqué lorsque la réponse est renvoyée par la méthode WebService est `PendingCall.onResult()` ou `onFault()`. En identifiant de manière unique vos objets de rappel, vous pouvez gérer plusieurs rappels `onResult`, comme dans l'exemple suivant :

```
monServiceWeb = new WebService("http://www.maSociété.com/monService.wsdl");
rappel1 = monServiceWeb.obtenirMétéo("02451");
rappel1.onResult = function(résultat)
{
  //faire quelque chose
}
```

```
rappel2 = monServiceWeb.météoDétaillée("02451");
rappel2.onResult = function(résultat)
{
  //faire autre chose
}
```

Description

Pour invoquer une opération de service web, invoquez-la comme une méthode directement disponible sur le service web. Par exemple, si votre service web possède la méthode `obtenirInfosSociété(symboleTéléscripteur)`, appelez alors :

```
monObjetDeRappel.monservice.obtenirInfosSociété(symboleTéléscripteur);
```

Toutes les invocations sont asynchrones et renvoient un objet de rappel, du type `PendingCall`.

WebServiceConnector (Flash Professionnel uniquement)

Le composant WebServiceConnector permet d'accéder aux méthodes distantes présentées par un serveur utilisant le protocole standard SOAP (Simple Object Access Protocol). Un service web peut accepter des paramètres et renvoyer un résultat. A l'aide de l'outil de programmation Flash MX Professionnel 2004 et du composant WebServiceConnector, vous pouvez lancer des analyses par introspection, accéder aux données d'un serveur web distant et lier ces données à votre application Flash. Une seule occurrence du composant WebServiceConnector peut être utilisée pour appeler plusieurs fois la même opération. Vous devez utiliser une occurrence différente d'un composant WebServiceConnector pour chacune des opérations que vous souhaitez appeler.

Un service web définit les méthodes (parfois désignées sous le nom d'« opérations ») pouvant être utilisées par un fichier XML à l'aide du format WSDL (Web Service Description Language). Le fichier WSDL énumère les opérations, paramètres et résultats (appelés le schéma) présentés par le service web.

Les fichiers WSDL sont accessibles par une URL. Dans Flash MX Professionnel 2004, vous pouvez visualiser le schéma de n'importe quel service web en entrant l'URL de son fichier WSDL dans le panneau Services Web. Une fois le fichier WSDL identifié, vous pouvez utiliser le service web pour toutes les applications que vous créez.

Seul l'auteur du fichier WSDL peut modifier le fichier WSDL ou le paramètre opération. Lorsque l'auteur modifie le fichier WSDL, les schémas de paramètres et de résultats sont mis à jour en fonction de ces modifications. Ces modifications écrasent tout changement apporté au schéma par le développeur. Pour obtenir un fichier WSDL mis à jour, vous pouvez sélectionner Actualiser les services Web dans le menu du panneau Web Service.

Le composant WebServiceConnector et le composant XMLConnector implémentent l'API du composant RPC (Remote Procedure Call), un ensemble de méthodes, propriétés et événements qui définissent un moyen simple d'envoyer des paramètres vers une source de données externe et de recevoir des résultats de cette dernière.

Une seule occurrence du composant WebServiceConnector peut être utilisée pour appeler plusieurs fois la même opération. Vous devez utiliser une occurrence différente de WebServiceConnector pour chaque opération distincte que vous souhaitez appeler.

Un développeur peut modifier le schéma pour le personnaliser en vue d'une utilisation dans une application (par exemple, pour fournir des paramètres supplémentaires de formatage ou de validation). Consultez la rubrique « Utilisation des schémas dans l'onglet Schémas (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Utilisation du composant WebServiceConnector (Flash Professionnel uniquement)

Vous pouvez utiliser le composant WebServiceConnector pour vous connecter à un service web et pour rendre les propriétés du service web disponibles pour effectuer la liaison avec les propriétés des composants de l'interface utilisateur de votre application. Pour vous connecter à un service web, vous devez tout d'abord entrer l'URL du service web concerné. Si le composant WebServiceConnector apparaît sur la scène au cours de la programmation de l'application, il n'est pas visible dans l'application.

Vous pouvez entrer l'URL d'un service web dans le panneau Inspecteur de composants ou dans le panneau Services Web.Consultez « Composant WebServiceConnector », dans le guide Utilisation de Flash de l'aide.

Pour plus d'informations sur l'utilisation du composant WebServiceConnector, consultez « Liaison des données (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Paramètres du composant WebServiceConnector

Les paramètres suivants sont des paramètres de programmation que vous pouvez définir pour chaque occurrence de composant WebServiceConnector, dans l'onglet Paramètres du panneau Inspecteur de composants :

Le paramètre `multipleSimultaneousAllowed` (type Booléen) indique si plusieurs appels peuvent avoir lieu en même temps ; la valeur par défaut est définie sur false. Si la valeur par défaut est false, alors la fonction `trigger()` n'effectue aucun appel si un appel est déjà en cours. Un événement d'état est provoqué, avec le code CallAlreadyInProgress. Si la valeur est true, alors l'appel a lieu.

Le paramètre `operation` (type Chaîne) correspond au nom d'une opération qui apparaît dans le port SOAP d'un fichier WSDL.

Le paramètre `suppressInvalidCalls` (type Booléen) indique s'il faut supprimer un appel lorsque les paramètres ne sont pas valides ; la valeur par défaut est définie sur false. Si la valeur est true, alors la fonction `trigger()` n'effectue aucun appel si les paramètres liés aux données ne réalisent pas la validation. Un événement d'état est émis, avec le code InvalidParams. Si la valeur est false, alors l'appel a lieu et utilise les données non valides le cas échéant.

Le paramètre `WSDLURL` (type Chaîne) correspond à l'URL du fichier WSDL qui définit l'opération du service web. Lorsque vous définissez l'URL au cours de la programmation, le fichier WSDL est immédiatement récupéré et analysé. Les paramètres qui en découlent et les informations de résultats sont visibles dans l'onglet Schéma du panneau Inspecteur de composants. La description du service est ajoutée dans le panneau Service Web. Pour obtenir un exemple, consultez l'adresse www.xmethods.net/sd/2001/TemperatureService.wsdl.

Classe WebServiceConnector (Flash Professionnel uniquement)

Héritage RPC > WebServiceConnector

Nom de classe ActionScript mx.data.components.WebServiceConnector

Propriétés de la classe WebServiceConnector

Propriété	Description
WebServiceConnector.multipleSimultaneousAllowed	Indique si plusieurs appels peuvent être effectués simultanément.
WebServiceConnector.multipleSimultaneousAllowed	Indique le nom d'une opération qui apparaît dans le port SOAP d'un fichier WSDL.
WebServiceConnector.params	Spécifie les données qui seront envoyées au serveur lors de l'exécution de la prochaine opération `trigger()`.
WebServiceConnector.results	Identifie les données reçues du serveur suite à l'opération `trigger()`.
WebServiceConnector.suppressInvalidCalls	Indique si un appel doit être supprimé lorsque ses paramètres ne sont pas valides.
WebServiceConnector.timeout	Détermine une période de temps (en secondes) au cours de laquelle la connexion au service web échoue si les résultats ne sont pas renvoyés.
WebServiceConnector.WSDLURL	Détermine l'URL du fichier WSDL définissant l'opération du service web.

Méthodes de la classe WebServiceConnector

Méthode	Description
WebServiceConnector.trigger()	Lance un appel de procédure à distance.

Evénements de la classe WebServiceConnector

Evénement	Description
WebServiceConnector.result	Diffusé lorsqu'un appel vers un service web est réussi.
WebServiceConnector.send	Diffusé lorsque la fonction `trigger()` est en cours, une fois que les données de paramètres ont été réunies mais avant que les données ne soient validées et que l'appel vers le service web ne soit initié.
WebServiceConnector.status	Diffusé lors du lancement d'un appel vers le service web, pour informer l'utilisateur de l'état de l'opération.

WebServiceConnector.multipleSimultaneousAllowed

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.multipleSimultaneousAllowed;
```

Description

Propriété : indique si plusieurs appels peuvent être effectués simultanément. Si elle est définie sur false, la fonction `trigger()` lance un appel si un autre appel est déjà en cours. Un événement `status` est émis, avec le code `CallAlreadyInProgress`. Si la valeur est true, alors l'appel a lieu.

Lorsque plusieurs appels ont lieu simultanément, il n'est pas garanti qu'ils se termineront dans l'ordre dans lequel ils ont été déclenchés. Flash Player peut aussi limiter le nombre d'opérations réseau simultanées. Cette limite dépend de la version et de la plate-forme.

Exemple

L'exemple suivant permet plusieurs appels simultanés vers `monUrlXml` :

```
monUrlXml.multipleSimultaneousAllowed = true;
```

WebServiceConnector.operation

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.operation;
```

Description

Propriété : nom d'une opération qui apparaît dans le port SOAP d'un fichier WSDL.

WebServiceConnector.params

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.params;
```

Description

Propriété : détermine les données qui seront envoyées au serveur lors de l'exécution de la prochaine opération `trigger()`. Le type de données est fixé par la description WSDL du service web.

Lorsque vous appelez des méthodes de service web, le type de données de la propriété params doit être un objet ou un tableau ActionScript :

Si le service web est au format document, alors le type de données de `params` est un document XML.

Si vous utilisez le panneau Inspecteur des propriétés ou Inspecteur de composants pour définir les paramètres WSDLURL et operation lors de la programmation, vous pouvez fournir `params` sous forme de tableau de paramètres, dans l'ordre requis par la méthode du service web ([1, "bonjour", 2432], par exemple).

Exemple

L'exemple suivant définit la propriété params d'un composant de service web appelé `wsc` :

```
wsc.params = [param_txt.text];
```

WebServiceConnector.result

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.addEventListener("result", monObjetDécoute);
```

Description

Evénement : diffusé lorsqu'une opération RPC réussit.

Le paramètre du gestionnaire d'événement est un objet comportant les champs suivants :

- `type` : la chaîne "result"
- `cible` : référence à l'objet qui a provoqué l'événement (un composant WebServiceConnector, par exemple).

Vous pouvez récupérer la valeur réelle du résultat à l'aide de la propriété `results`.

Exemple

L'exemple suivant définit une fonction `résultats` pour l'événement `result` et affecte la fonction au gestionnaire d'événement `addEventListener` :

```
var résultats = function (ev) {
trace(ev.target.results);
};
wsc.addEventListener("result", résultats);
```

WebServiceConnector.results

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.results;
```

Description

Propriété : identifie les données reçues en provenance du serveur suite à l'opération `trigger()`. Chaque composant RPC définit la manière dont les données sont récupérées et quels sont les types valides. Ces données apparaissent lorsque l'opération RPC a réussi, comme indiqué par l'événement `result`. Elles sont disponibles jusqu'à la purge du composant ou jusqu'à l'opération RPC suivante.

Les données renvoyées peuvent être très volumineuses. Il existe deux manières de les gérer :

- Sélectionnez un clip, un scénario ou un écran approprié en tant que parent du composant RPC. Lorsque le parent ne sera plus disponible, le stockage de ce composant pourra être utilisé pour collecter divers éléments.
- Vous pouvez affecter à tout moment la valeur null à cette propriété à l'aide d'ActionScript.

WebServiceConnector.send

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.addEventListener("send", monObjetDécoute);
```

Description

Evénement : diffusé pendant le traitement d'une opération `trigger()`, après la collecte des paramètres, mais avant la validation des données et le lancement de l'appel à distance. Cet événement permet d'insérer du code destiné à modifier les paramètres avant l'appel.

Le paramètre du gestionnaire d'événement est un objet comportant les champs suivants :

- `type` : la chaîne "send"
- `cible` : référence à l'objet qui a provoqué l'événement (un composant WebServiceConnector, par exemple).

Vous pouvez récupérer ou modifier les valeurs réelles des paramètres à l'aide de la propriété `params`.

Exemple

L'exemple suivant définit une fonction `fonctionEnvoyer` pour l'événement `send` et affecte la fonction au gestionnaire d'événement `addEventListener` :

```
var fonctionEnvoyer = function (sendEnv) {
sendEnv.target.params = [newParam_txt.text];
};
wsc.addEventListener("send", fonctionEnvoyer);
```

WebServiceConnector.status

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.addEventListener("status", monObjetDécoute);
```

Description

Evénement : diffusé lorsqu'un appel de procédure à distance est lancé, pour informer l'utilisateur de l'état de l'opération.

Le paramètre du gestionnaire d'événement est un objet comportant les champs suivants :

- `type` : la chaîne "status"
- `cible` : référence à l'objet qui a provoqué l'événement (un composant WebServiceConnector, par exemple).
- `code` : une chaîne donnant le nom de la condition spécifique qui s'est produite.
- `data` : un objet dont le contenu dépend du code.

Les codes et les données associées applicables à l'événement status sont les suivants :

Code	Données	Description
StatusChange	{callsInProgress:nnn}	Cet événement se produit chaque fois qu'un appel à un service web est lancé ou se termine. L'élément "nnn" correspond au nombre d'appels en cours.

Code	Données	Description
CallAlreadyInProgress	aucune donnée	Cet événement se produit si (a) la fonction `trigger()` est appelée, si (b) multipleSimultaneousAllowed est false et si (c) un appel est déjà en cours. Une fois que l'événement s'est produit, l'appel tenté est considéré comme terminé ; aucun événement "result" ou "send" ne se produit.
InvalidParams	aucune donnée	Cet événement se produit si la fonction `trigger()` a déterminé que la propriété "params" ne contenait pas de données valides. Si la propriété "suppressInvalidCalls" est définie sur true, l'appel tenté est considéré comme terminé et aucun événement "result" ou "send" ne se produit.

Voici les erreurs possibles du service web :

faultcode	faultstring	detail
Timeout	Timeout while calling method xxx	þ
MustUnderstand	No callback for header xxx	þ
Server.Connection	Unable to connect to endpoint : xxx	þ
VersionMismatch	Request implements version : xxx Response implements version yyy	þ
Client.Disconnected	Could not load WSDL	Unable to load WSDL, if currently online, please verify the URI and/or format of the WSDL xxx
Server	Faulty WSDL format	Definitions must be the first element in a WSDL document
Server.NoServicesInWSDL	Could not load WSDL	No elements found in WSDL at xxx
WSDL.UnrecognizedNamespace	The WSDL parser had no registered document for the namespace xxxx	þ
WSDL.UnrecognizedBindingName	The WSDL parser couldn't find a binding named xxx in namespace yyy	þ
WSDL.UnrecognizedPortTypeName	The WSDL parser couldn't find a portType named xxx in namespace yyy	þ

faultcode	faultstring	detail
WSDL.UnrecognizedMessageName	The WSDL parser couldn't find a message named xxx in namespace yyy	þ
WSDL.BadElement	Element xxx not resolvable	þ
WSDL.BadType	Type xxx not resolvable	þ
Client.NoSuchMethod	Couldn't find method 'xxx' in service	þ
yyy	yyy - errors reported from server, this depends on which server you talk to	þ
No.WSDLURL.Defined	the WebServiceConnector component had no WSDL URL defined	þ
Unknown.Call.Failure	WebService invocation failed for unknown reasons	þ
Client.Disconnected	Could not load imported schema	Unable to load schema; if currently online, please verify the URI and/or format of the schema at (XXXX)

Exemple

L'exemple suivant définit une fonction `fonctionEtat` pour l'événement `status` et affecte la fonction au gestionnaire d'événement `addEventListener` :

```
var fonctionEtat = function (stat) {
trace(stat.code);
trace(stat.data.faultcode);
trace(stat.data.faultstring);
};
wsc.addEventListener("status", fonctionEtat);
```

WebServiceConnector.suppressInvalidCalls

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

occurrenceDeComposant`.suppressInvalidCalls;`

Description

Propriété : indique si un appel doit être supprimé lorsque ses paramètres ne sont pas valides. Si la valeur est `true`, alors la fonction `trigger()` n'effectue aucun appel si les paramètres liés ne sont pas validés. Un événement "status" est émis, avec le code `InvalidParams`. Si la valeur est `false`, alors l'appel a lieu et utilise les données non valides le cas échéant.

WebServiceConnector.timeout

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.timeout;
```

Description

Propriété : période de temps en secondes au cours de laquelle la connexion au service web échoue si les résultats ne sont pas renvoyés. Un événement `status` (hérité du composant RPC) est provoqué, avec le code `WebServiceFault` et le code d'erreur `Timeout`.

WebServiceConnector.trigger()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.trigger();
```

Description

Méthode : initie un appel vers un service web. Chaque service web définit exactement ce que l'appel implique. Si l'opération réussit, les résultats de l'opération apparaissent dans la propriété `results` du service web.

La méthode `trigger()` effectue les opérations suivantes :

1. Si des données sont liées à la propriété `params`, la méthode exécute toutes les liaisons afin de garantir que les données disponibles sont à jour. Cela provoque également la validation des données.
2. Si les données ne sont pas valides alors que le paramètre `suppressInvalidCalls` est défini sur `true`, l'opération est interrompue.
3. Si l'opération continue, l'événement send se produit.
4. L'appel distant est initié à l'aide de la méthode de connexion indiquée (HTTP, par exemple).

WebServiceConnector.WSDLURL

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.WSDLURL;
```

Description

Propriété : l'URL du fichier WSDL qui définit l'opération du service web. Lorsque vous définissez l'URL au cours de la programmation, le fichier WSDL est immédiatement récupéré et analysé. Les paramètres et les résultats qui en découlent apparaissent dans l'onglet Schéma du panneau Inspecteur de composants. La description du service apparaît également dans le panneau Services Web.

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.

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.

Utilisation du composant Window

Vous pouvez utiliser une fenêtre dans une application dès que vous avez besoin de présenter à l'utilisateur une information ou un choix prioritaire par rapport au reste de l'application. Par exemple, si vous souhaitez que l'utilisateur renseigne une fenêtre de connexion ou une fenêtre qui permet de modifier un mot de passe et d'en confirmer un nouveau.

Il existe plusieurs manières d'ajouter une fenêtre à une application. Vous pouvez faire glisser une fenêtre du panneau Composants jusqu'à la scène. Vous pouvez également appeler `createClassObject()` (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*.

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, l'ordre de focus dépend du contenu de la fenêtre et des autres composants du document. Pour plus d'informations sur le contrôle du focus, consultez *Création de la navigation personnalisée du focus*, page 27 ou *Classe FocusManager*, page 287.

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 Window, effectuez les opérations suivantes :

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**.

   Il s'agit du futur contenu de la fenêtre. Le contenu doit être aligné à 0,0 car il est positionné dans le coin supérieur gauche de la fenêtre.
2. Dans la bibliothèque, sélectionnez le clip FormulaireMotDePasse et choisissez Liaison dans le menu d'options.
3. Activez l'option Exporter pour ActionScript.

   L'identifiant de liaison **FormulaireMotDePasse** est entré automatiquement dans la zone Identifiant.
4. Entrez **mx.core.View** dans le champ de classe et cliquez sur OK.
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 si elle ne l'est pas déjà.

8 Faites glisser un composant Button du panneau Composants vers la scène ; dans l'Inspecteur des propriétés, donnez-lui le nom d'occurrence **bouton**.

9 Ouvrez le panneau Actions et entrez le gestionnaire de clic suivant sur l'image 1 :

```
écouteurDeBouton = new Object();
écouteurDeBouton.click = function(){
  mx.managers.PopUpManager.createPopUp(_root, mx.containers.Window, true, {
  title:"Modifier mon mot de passe", contentPath:"FormulaireMotDePasse" });
  maFenêtre.setSize(240 110);
}
bouton.addEventListener("click", écouteurDeBouton);
```

Ce gestionnaire appelle `PopUpManager.createPopUp()` pour instancier un composant Window avec la barre de titre « Modifier mon mot de passe », qui affiche le contenu du clip FormulaireMotDePasse lorsque vous cliquez sur le bouton. Pour que la fenêtre se ferme lorsque vous cliquez sur OK ou Annuler, vous devez rédiger un autre gestionnaire.

Personnalisation du composant Window

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

La modification de la taille de la fenêtre ne change pas la taille du bouton ou du titre. Le titre est aligné sur la gauche et la barre de fermeture sur la droite.

Utilisation de styles avec le composant Window

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.

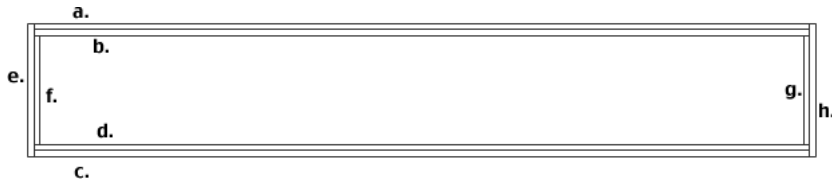
Utilisation d'enveloppes avec le composant Window

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	Lettre
`borderColor`	a
`highlightColor`	b
`borderColor`	c
`shadowColor`	d
`borderCapColor`	e

Styles RectBorder	Lettre
shadowCapColor	f
shadowCapColor	g
borderCapColor	h

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`.
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

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 des classes *UIObject*, *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 des classes *UIObject*, *UIComponent* et ScrollView.

Evénements de la classe Window

Evénement	Description
`Window.click`	Diffusé lorsque le bouton de fermeture est relâché.
`Window.mouseDownOutside`	Diffusé lorsque l'utilisateur clique avec la souris en dehors de la fenêtre modale.

Hérite de tous les événements des classes *UIObject*, *UIComponent*, View et ScrollView.

Window.click

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

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

Usage 2 :

```
objetDécoute = new Object();
objetDécoute.click = function(objetEvt){
   ...
}
occurrenceFenêtre.addEventListener("click", objetDécoute)
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque l'utilisateur clique sur le bouton Fermer (puis relâche le bouton de la souris).

Le premier exemple d'utilisation fait appel 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 (dans ce cas, `click`) qui est géré par une fonction (appelée aussi *gestionnaire*) sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

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);
```

Voir aussi

`UIEventDispatcher.addEventListener()`, `Window.closeButton`

Window.closeButton

Disponibilité

Flash Player 6 version 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});
```

Voir aussi

`Window.click`, `PopUpManager.createPopUp()`

Window.content

Disponibilité

Flash Player 6 version 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 version 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 version 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 version 79.

Edition

Flash MX 2004.

Usage

Usage 1 :

```
on(mouseDownOutside){
  ...
}
```

Usage 2 :

```
objetDécoute = new Object();
listenerObject.mouseDownOutside = function(objetEvt){
  ...
}
occurrenceFenêtre.addEventListener("mouseDownOutside", objetDécoute)
```

Description

Evénement : diffusé à tous les écouteurs enregistrés lorsque l'utilisateur clique sur le bouton de la souris (puis le relâche) en dehors de la fenêtre modale. Cet événement est rarement utilisé, mais vous pouvez vous en servir pour fermer une fenêtre si l'utilisateur essaie d'interagir avec un élément situé à l'extérieur de cette fenêtre.

Le premier exemple d'utilisation fait appel 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 (dans ce cas, `mouseDownOutside`) qui est géré par une fonction (appelée aussi *gestionnaire*) sur 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 transmet automatiquement un objet événement (*`objetEvt`*) 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 événement, consultez *Objets événement*, page 592.

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);
```

Voir aussi

`UIEventDispatcher.addEventListener()`

Window.title

Disponibilité

Flash Player 6 version 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 version 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 29.

Composant XMLConnector (Flash Professionnel uniquement)

Le composant XMLConnector est un composant Flash MX 2004 v2 dont le but est de lire ou rédiger des documents XML à l'aide d'opérations HTTP `get` ou HTTP `post`. Il agit en tant que connecteur entre les autres composants et des sources de données XML externes. Le composant XMLConnector communique avec les composants de votre application en utilisant soit les fonctions de liaison de données disponibles dans l'environnement de programmation Flash MX Professionnel 2004, soit du code ActionScript. Le composant XMLConnector possède des propriétés, des méthodes et des événements, mais n'est pas visible à l'exécution.

Les composants XMLConnector et WebServiceConnector implémentent l'API du composant RPC (Remote Procedure Call), un ensemble de méthodes, de propriétés et d'événements qui définissent un moyen simple d'envoyer des paramètres vers une source de données externe et de recevoir les résultats de celle-ci.

Utilisation du composant XMLConnector (Flash Professionnel uniquement)

Le composant XMLConnector permet à l'application d'accéder à toute source de données externe renvoyant ou recevant des données XML via HTTP. Le moyen le plus simple de se connecter à une source de données XML externe et d'utiliser les paramètres et les résultats de cette source de données consiste à spécifier un *schéma,* c'est-à-dire la structure du document XML qui identifie les éléments de données du document sur lesquels vous pouvez créer des liaisons.

Le schéma apparaît dans l'onglet Schéma du panneau Inspecteur de composants. Le schéma identifie les champs du document XML que vous pouvez lier aux propriétés des composants de l'interface utilisateur de votre application. Vous pouvez créer manuellement le schéma, par l'intermédiaire du panneau Inspecteur de composants, ou utiliser l'environnement de programmation pour en créer un automatiquement.

Remarque : L'environnement de programmation accepte une copie du document XML externe auquel vous vous connectez en tant que modèle de schéma. Si vous connaissez la programmation XML, vous pouvez créer un fichier XML à utiliser pour générer le schéma.

Bien que le composant XMLConnector dispose de propriétés et d'événements (comme d'autres composants), il n'a pas d'apparence visuelle à l'exécution. Pour plus d'informations sur l'utilisation du composant XMLConnector, consultez « Composant XMLConnector (Flash Professionnel uniquement) », dans le guide d'Utilisation de Flash de l'aide.

Paramètres du composant XMLConnector

Les paramètres suivants sont des paramètres de programmation que vous pouvez définir pour chaque occurrence de composant XMLConnector, dans l'onglet Paramètres du panneau Inspecteur de composants :

Le paramètre `direction` (Enumération) indique si les données sont envoyées, reçues, ou les deux.

Le paramètre `ignoreWhite` (type Booléen) lorsque cette valeur est définie sur true, ignore les espaces lorsque le XML est récupéré.

Le paramètre `multipleSimultaneousAllowed` (type Booléen) indique si plusieurs appels peuvent avoir lieu en même temps ; la valeur par défaut est définie sur false.

Le paramètre `suppressInvalidCalls` (type Booléen) indique s'il faut supprimer un appel lorsque les paramètres ne sont pas valides ; la valeur par défaut est définie sur false.

Le paramètre `URL` (type de chaîne) correspond à l'URL du document XML externe utilisé dans les opérations HTTP.

Classe XMLConnector (Flash Professionnel uniquement)

Héritage RPC > XMLConnector

Nom de classe ActionScript mx.data.components.XMLConnector

Propriétés de la classe XMLConnector

Propriété	Description
`XMLConnector.direction`	Indique si les données sont envoyées, reçues ou les deux.
`XMLConnector.multipleSimultaneousAllowed`	Indique si plusieurs appels peuvent être effectués simultanément.
`XMLConnector.params`	Spécifie les données qui seront envoyées au serveur lors de l'exécution de la prochaine opération `trigger()`.
`XMLConnector.results`	Identifie les données reçues du serveur suite à l'opération `trigger()`.
`XMLConnector.suppressInvalidCalls`	Indique si un appel doit être supprimé lorsque ses paramètres ne sont pas valides.
`XMLConnector.URL`	L'URL utilisé par le composant dans les opérations HTTP.

Méthodes de la classe XMLConnector

Méthode	Description
`XMLConnector.trigger()`	Lance un appel de procédure à distance.

Evénements de la classe XMLConnector

Evénement	Description
`XMLConnector.result`	Diffusé lorsqu'un appel de procédure à distance (RPC) s'achève avec succès.
`XMLConnector.send`	Diffusé lorsque la fonction `trigger()` est en cours, après la collecte des paramètres, mais avant la validation des données et le lancement de l'appel à distance.
`XMLConnector.status`	Diffusé lorsqu'un appel de procédure à distance est lancé, pour informer l'utilisateur de l'état de l'opération.

XMLConnector.direction

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.direction;
```

Description

Propriété : indique si les données sont envoyées, reçues ou les deux. Les valeurs sont les suivantes :

- `send` Les données XML pour la propriété `params` sont envoyées via HTTP POST à l'URL pour le document XML. Toutes les données renvoyées sont ignorées. La propriété `results` n'est pas définie et il n'y a aucun événement `result`.

  Remarque : La propriété params et l'événement de résultat sont hérités de l'API du composant RPC.

- `receive` Aucune donnée params n'est envoyée à l'URL. Vous pouvez accéder à l'URL du document XML via HTTP GET et l'URL doit envoyer des données XML valides.
- `send/receive` Les données Params sont envoyées vers l'URL qui doit envoyer des données XML valides.

Exemple

L'exemple suivant définit la direction du paramètre `receive` pour le document `mesparamètres.xml` :

```
monConnecteurXML.direction = "receive";
monConnecteurXML.URL = "mesparamètres.xml";
monConnecteurXML.trigger();
```

XMLConnector.multipleSimultaneousAllowed

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.multipleSimultaneousAllowed;
```

Description

Propriété : indique si plusieurs appels peuvent être effectués simultanément. Si elle est définie sur false, la fonction `trigger()` lance un appel si un autre appel est déjà en cours. Un événement `status` est émis, avec le code `CallAlreadyInProgress`. Si la valeur est true, alors l'appel a lieu.

Lorsque plusieurs appels ont lieu simultanément, il n'est pas garanti qu'ils se termineront dans l'ordre dans lequel ils ont été déclenchés. Flash Player peut aussi limiter le nombre d'opérations réseau simultanées. Cette limite dépend de la version et de la plate-forme.

XMLConnector.params

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.params;
```

Description

Propriété : détermine les données qui seront envoyées au serveur lors de l'exécution de la prochaine opération `trigger()`. Chaque composant RPC définit la manière dont les données sont utilisées et quels sont les types valides.

Exemple

L'exemple suivant définit les params `name` et `city` pour `monConnecteurXML` :

```
monConnecteurXML.params = new XML("<mondoc><name>Bob</name><city>Oakland</
  city></mondoc>");
```

XMLConnector.result

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.addEventListener("result", monObjetDécoute);
```

Description

Evénement : diffusé en cas de succès d'une opération d'appel de procédure à distance.

Le paramètre du gestionnaire d'événement est un objet comportant les champs suivants :

- `type` : la chaîne "result".
- `cible` : référence à l'objet qui a provoqué l'événement (un composant WebServiceConnector, par exemple).

Vous pouvez récupérer la valeur réelle du résultat à l'aide de la propriété `results`.

Exemple

L'exemple suivant définit une fonction `résultats` pour l'événement `result` et affecte la fonction au gestionnaire d'événement `addEventListener` :

```
var résultats = function (ev) {
trace(ev.target.results);
};
xcon.addEventListener("result", res);
```

XMLConnector.results

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.results;
```

Description

Propriété : identifie les données reçues en provenance du serveur suite à l'opération `trigger()`. Chaque composant RPC définit la manière dont les données sont récupérées et quels sont les types valides. Ces données apparaissent lorsque l'opération RPC a réussi, comme indiqué par l'événement `result`. Elles sont disponibles jusqu'à la purge du composant ou jusqu'à l'opération RPC suivante.

Les données renvoyées peuvent être très volumineuses. Il existe deux manières de les gérer :

- Sélectionnez un clip, un scénario ou un écran approprié en tant que parent du composant RPC. Lorsque le parent ne sera plus disponible, le stockage de ce composant pourra être utilisé pour collecter divers éléments.
- Vous pouvez affecter à tout moment la valeur null à cette propriété à l'aide d'ActionScript.

Exemple

L'exemple suivant présente la propriété `results` pour `monConnecteurXML` :

```
trace(monConnecteurXML.results);
```

XMLConnector.send

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.addEventListener("send", monObjetDécoute);
```

Description

Evénement : diffusé pendant le traitement d'une opération `trigger()`, après la collecte des paramètres, mais avant la validation des données et le lancement de l'appel à distance. Cet événement permet d'insérer du code destiné à modifier les paramètres avant l'appel.

Le paramètre du gestionnaire d'événement est un objet comportant les champs suivants :

- `type` : la chaîne "send".
- `cible` : référence à l'objet qui a provoqué l'événement (un composant WebServiceConnector, par exemple).

Vous pouvez récupérer ou modifier les valeurs réelles des paramètres à l'aide de la propriété `params`.

Exemple

L'exemple suivant définit une fonction `fonctionEnvoyer` pour l'événement `send` et affecte la fonction au gestionnaire d'événement `addEventListener` :

```
var fonctionEnvoyer = function (sendEnv) {
sendEnv.target.params = [newParam_txt.text];
};
xcon.addEventListener("send", fonctionEnvoyer);
```

XMLConnector.status

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.addEventListener("status", monObjetDécoute);
```

Description

Evénement : diffusé lorsqu'un appel de procédure à distance est lancé, pour informer l'utilisateur de l'état de l'opération.

Le paramètre du gestionnaire d'événement est un objet comportant les champs suivants :

- `type` : la chaîne "status".

- `cible` : référence à l'objet qui a provoqué l'événement (un composant WebServiceConnector, par exemple).
- `code` : une chaîne donnant le nom de la condition spécifique qui s'est produite.
- `data` : un objet dont le contenu dépend du code.

Le champ de code pour l'événement status est défini sur Fault si des problèmes surviennent lors de l'appel, comme suit :

Code	Données	Description
Fault	{faultcode: code, faultstring: string, detail: detail, element: element, faultactor: actor}	Cet événement est provoqué si d'autres problèmes surviennent lors du traitement de l'appel. Les données constituent un objet SOAPFault. Une fois que l'événement s'est produit, l'appel tenté est considéré comme terminé ; aucun événement "result" ou "send" ne se produit.

Les erreurs suivantes sont les erreurs qui peuvent se produire avec l'événement status :

Code d'erreur	Chaîne	Remarques
`XMLConnector.Not.XML`	params is not an XML object	Les données params doivent être un objet XML d'actionscript.
`XMLConnector.Parse.Error`	params had XML parsing error NN	La propriété "status" de l'objet XML params avaient une valeur NN non nulle. Consultez les informations relatives à XML.status dans l'aide de Flash pour voir les erreurs NN possibles.
`XMLConnector.No.Data.Received`	no data was received from the server	RESTRICTION : en raison des limitations propres au navigateur, ce message peut signifier que (a) l'URL du serveur n'était pas valide, ne répondait pas ou a renvoyé un code d'erreur HTTP ; ou (b) que la demande du serveur a réussi mais que la réponse contenait 0 octets de données. Pour contourner ce problème, il est recommandé de créer votre application de façon à ce que le serveur ne renvoie JAMAIS 0 octets de données. Si le message « XMLConnector.No.Data.Received » s'affiche, vous êtes alors certain qu'il s'est produit une erreur au niveau du serveur et vous pouvez ainsi en informer l'utilisateur final.

Code d'erreur	Chaîne	Remarques
XMLConnector.Results.Parse.Error	received data had an XML parsing error NN	Le code XML reçu n'était pas valide, comme indiqué par le programme d'analyse XML intégré de Flash Player. Consultez les informations relatives à XML.status dans l'aide de Flash pour voir les erreurs NN possibles.
XMLConnector.Params.Missing	Direction is 'send' or 'send/receive', but params are null.	þ

Exemple

L'exemple suivant définit une fonction `fonctionEtat` pour l'événement `status` et affecte la fonction au gestionnaire d'événement `addEventListener` :

```
var fonctionEtat = function (stat) {
trace(stat.code);
trace(stat.data.faultcode);
trace(stat.data.faultstring);
};
xcon.addEventListener("status", fonctionEtat);
```

XMLConnector.suppressInvalidCalls

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

occurrenceDeComposant.suppressInvalidCalls;

Description

Propriété : indique si un appel doit être supprimé lorsque ses paramètres ne sont pas valides. Si la valeur est `true`, alors la fonction `trigger()` n'effectue aucun appel si les paramètres liés ne sont pas validés. Un événement "status" est émis, avec le code `InvalidParams`. Si la valeur est `false`, alors l'appel a lieu et utilise les données non valides le cas échéant.

XMLConnector.trigger()

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

occurrenceDeComposant.trigger();

Description

Méthode : initie un appel RPC. Chaque composant RPC définit précisément ce que cela implique. Si l'opération réussit, ses résultats apparaissent dans la propriété `results` du composant RPC.

La méthode `trigger()` effectue les opérations suivantes :

1. Si des données sont liées à la propriété `params`, la méthode exécute toutes les liaisons afin de garantir que les données disponibles sont à jour. Cela provoque également la validation des données.
2. Si les données ne sont pas valides alors que le paramètre `suppressInvalidCalls` est défini sur `true`, l'opération est interrompue.
3. Si l'opération continue, l'événement send se produit.
4. L'appel distant est initié à l'aide de la méthode de connexion indiquée (HTTP, par exemple).

XMLConnector.URL

Disponibilité

Flash Player 6 version 79.

Edition

Flash MX Professionnel 2004.

Usage

```
occurrenceDeComposant.URL;
```

Description

Propriété : l'URL que ce composant utilise lors des opérations HTTP. Il peut s'agir d'une URL absolue ou relative. L'URL est soumise à l'ensemble des protections de sécurité standard Flash Player.

Composant XUpdateResolver (Flash Professionnel uniquement)

Les composants Resolver s'utilisent en association avec le composant DataSet (partie de la fonctionnalité de gestion des données dans l'architecture de données Macromedia Flash). Les composants Resolver vous permettent de convertir les modifications apportées aux données de votre application dans un format adapté à la source de données externe que vous mettez à jour. Ils ne sont pas visibles à l'exécution.

Si vous utilisez un composant DataSet dans votre application, il génère un jeu optimisé d'instructions (DeltaPacket) décrivant les modifications apportées aux données à l'exécution. Ce jeu d'instructions est converti au format approprié (paquet de mise à jour) par les composants Resolver. Lorsqu'une mise à jour est envoyée au serveur, celui-ci envoie une réponse (paquet de résultats) contenant d'autres mises à jour ou des erreurs résultant de l'opération de mise à jour. Les composants Resolver peuvent reconvertir ces informations en DeltaPacket pour l'appliquer au composant DataSet afin que ce dernier reste synchronisé avec la source de données externe. Les composants Resolver vous permettent de maintenir votre application synchronisée avec une source de données externe sans écrire de code ActionScript supplémentaire.

XUpdate est un standard pour la description des modifications apportées à un document XML et il est pris en charge par un certain nombre de bases de données XML, telles que Xindice ou XHive. Le composant XUpdateResolver traduit DeltaPacket en instructions XUpdate. Une source de données externe peut traiter ces instructions XUpdates. Le document XML contient les informations et le formatage nécessaires à la mise à jour de toute base de données standard XUpdate.

Pour plus d'informations sur le modèle de spécification du langage XUpdate, consultez la page www.xmldb.org/xupdate/xupdate-wd.html. Un composant Resolver parallèle, RDBMSResolver (consultez *Composant RDBMSResolver (Flash Professionnel uniquement)*, page 461), permet le renvoi des données vers un serveur basé sur XML. Pour plus d'informations sur les composants DataSet, consultez *Composant DataSet (Flash Professionnel uniquement)*, page 207. Pour plus d'informations sur les connecteurs, consultez *WebServiceConnector (Flash Professionnel uniquement)*, page 636 et *Composant XMLConnector (Flash Professionnel uniquement)*, page 657. Pour plus d'informations sur l'architecture de données Flash, consultez « Composants Resolver (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Remarque : Vous pouvez également utiliser le composant XUpdateResolver pour envoyer des mises à jour de données à toute source de données externe pouvant analyser le langage XUpdate (une page ASP, un servlet Java ou un composant ColdFusion, par exemple).

Les mises à jour à partir du composant XUpdateResolver sont envoyées sous la forme d'un paquet de données XUpdate, qui est communiqué à la base de données ou au serveur par l'intermédiaire d'un objet de connexion. Le paquet XUpdate est constitué d'un jeu optimisé d'instructions décrivant les insertions, les modifications et les suppressions effectuées sur le composant DataSet. Le composant Resolver reçoit un DeltaPacket d'un composant DataSet, envoie son propre paquet XUpdate vers un connecteur, reçoit les erreurs du serveur de la connexion et les communique en retour au composant DataSet. Tous ces processus utilisent les propriétés de liaison.

Utilisation du composant XUpdateResolver (Flash Professionnel uniquement)

Utilisez ce composant XUpdateResolver uniquement lorsque votre application Flash contient un composant DataSet et doit renvoyer une mise à jour vers la source de données. Ce composant résout les données que vous souhaitez renvoyer vers une source de données au format XML.

Pour plus d'informations sur l'utilisation du composant XUpdateResolver, consultez « Composants Resolver (Flash Professionnel uniquement) », dans le guide Utilisation de Flash de l'aide.

Paramètres du composant XUpdateResolver

includeDeltaPacketInfo *Booléen* ; si la valeur est définie sur `true`, XUpdate inclut des informations complémentaires du `deltaPacket` dans les attributs des nœuds XUpdate. Ces informations comprennent l'ID de transaction et l'ID d'opération.

L'exemple suivant montre comment le paquet de mise à jour est construit lorsque la valeur booléenne de cette propriété est définie sur `false` :

```
<xupdate:modifications
  version="1.0"
  xmlns:xupdate="http://www.xmldb.org/xupdate">

  <xupdate:remove select="/datapacket/row[@id='100']"/>
```

```
</xupdate:modifications>
```

L'exemple suivant montre comment le paquet de mise à jour est construit lorsque la valeur booléenne de cette propriété est définie sur `true` :

```
<xupdate:modifications
  version="1.0"
  xmlns:xupdate="http://www.xmldb.org/xupdate"
  transID="46386292065:Mer 25 juin 15:52:34 GMT-0700 2003">

  <xupdate:remove select="/datapacket/row[@id='100']" opId="0123456789"/>

</xupdate:modifications>
```

Propriétés du composant XUpdateResolver

Propriété	Description
`XUpdateResolver.deltaPacket`	Contient une description des modifications apportées au composant DataSet.
`XUpdateResolver.includeDeltaPacketInfo`	Comprend des informations supplémentaires du `deltaPacket` dans les attributs des nœuds XUpdate.
`XUpdateResolver.updateResults`	Décrit les résultats de la mise à jour.
`XUpdateResolver.xupdatePacket`	Contient la traduction XUpdate des modifications vers le composant DataSet.

Evénements du composant XUpdateResolver

Evénement	Description
`XUpdateResolver.beforeApplyUpdates`	Appelé par le composant Resolver pour apporter des modifications personnalisées immédiatement après la création du paquet XML et juste avant son envoi.
`XUpdateResolver.reconcileResults`	Appelé par le composant Resolver pour comparer deux paquets.

XUpdateResolver.beforeApplyUpdates

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

`donnéesResolve``.beforeApplyUpdates(`*`objetEvt`*`)`

Paramètres

objetEvt Objet événement Resolver ; décrit les personnalisations apportées au paquet XML avant l'envoi de la mise à jour à la base de données via le connecteur. Cet objet événement Resolver doit contenir les propriétés suivantes :

Propriété	Description
target	Objet ; resolver déclenchant l'événement.
type	Chaîne ; nom de l'événement.
updatePacket	Objet XML ; l'objet XML qui sera appliqué.

Renvoie

Aucun.

Description

Evénement : appelé par le composant Resolver pour apporter des modifications personnalisées immédiatement après la création du paquet XML pour un nouveau deltaPacket et juste avant l'envoi de ce paquet à l'aide de la liaison de données. Vous pouvez utiliser ce gestionnaire d'événement pour effectuer des modifications personnalisées dans le XML avant d'envoyer les données mises à jour vers un connecteur.

Exemple

L'exemple suivant ajoute les données d'authentification de l'utilisateur au paquet XML :

```
on (beforeApplyUpdates) {
   // ajouter les données d'authentification de l'utilisateur
   var infosUtilisateur = new XML( ""+getUserId()+" "+getPassword()+"" );
   xupdatePacket.firstChild.appendChild( infosUtilisateur );
}
```

XUpdateResolver.deltaPacket

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

donnéesResolve.deltaPacket

Description

Propriété : propriété de type deltaPacket qui reçoit un deltaPacket à traduire en xupdatePacket et produit un deltaPacket à partir de n'importe quels résultats serveur placés dans la propriété updateResults. Ce gestionnaire d'événement vous permet d'apporter des modifications personnalisées au paquet XML avant l'envoi des données mises à jour au connecteur.

Les messages dans la propriété `updateResults` sont considérés comme des erreurs. Cela signifie qu'un paquet delta contenant des messages est de nouveau ajouté au `deltaPacket` pour être renvoyé lors de la prochaine transmission du `deltaPacket` au serveur. Vous devez rédiger le code qui gère les deltas contenant des messages pour que les messages soient présentés à l'utilisateur et modifiés avant d'être ajoutés au `deltaPacket` suivant.

XUpdateResolver.includeDeltaPacketInfo

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

`resolveData``.includeDeltaPacketInfo`

Description

Propriété : propriété de type `booléen` qui, si elle est définie sur `true`, inclut des informations complémentaires du `deltaPacket` dans les attributs sur les nœuds XUpdate. Ces informations comprennent l'ID de transaction et l'ID d'opération.

Pour voir un exemple du XML obtenu, consultez *Paramètres du composant XUpdateResolver*, page 666.

XUpdateResolver.reconcileResults

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

`donnéesResolve``.reconcileResults(`*`objetEvt`*`)`

Paramètres

`objetEvt` Objet ResolverEvent ; décrit l'objet événement utilisé pour comparer deux `updatePacket`s. Cet objet événement Resolver doit contenir les propriétés suivantes :

Propriété	Description
`target`	Objet ; resolver déclenchant l'événement.
`type`	Chaîne ; nom de l'événement.

Renvoie

Aucun.

Description

Evénement : utilisez ce rappel pour insérer tout code après réception des résultats du serveur et juste avant la transmission, par la liaison de données, du `deltaPacket` contenant les résultats de l'opération. Il s'agit d'un bon emplacement pour insérer du code qui gère les messages en provenance du serveur.

Exemple

L'exemple suivant reconstitue deux `updatePacket`s et efface les mises à jour réussies :

```
on (reconcileResults) {
      // examiner les résultats
      if( examine( updateResults ))
        monEnsembleDeDonnées.purgeUpdates();
      else
       displayErrors( results );
}
```

XUpdateResolver.updateResults

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

`resolveData``.updateResults`

Description

Propriété : propriété de type `deltaPacket` qui contient les résultats d'une mise à jour renvoyés à partir du serveur à l'aide d'un connecteur. Utilisez cette propriété pour transmettre les erreurs et les données mises à jour à partir du serveur vers un composant DataSet ; par exemple, lorsque le serveur affecte de nouveaux ID pour un champ auto-affecté. Liez cette propriété à la propriété `Results` d'un connecteur de manière à ce qu'il puisse recevoir les résultats d'une mise à jour et renvoyer les résultats au composant DataSet.

Les messages dans la propriété `updateResults` sont considérés comme des erreurs. Cela signifie qu'un paquet delta contenant des messages est de nouveau ajouté au `deltaPacket` pour être renvoyé lors de la prochaine transmission du `deltaPacket` au serveur. Vous devez rédiger le code qui gère les deltas contenant des messages pour que les messages soient présentés à l'utilisateur et modifiés avant d'être ajoutés au `deltaPacket` suivant.

XUpdateResolver.xupdatePacket

Disponibilité

Flash Player 7.

Edition

Flash MX Professionnel 2004.

Utilisation

resolveData.xupdatePacket

Description

Propriété : la propriété de type `xml` contient la traduction XUpdate des modifications apportées au composant DataSet. Effectuez la liaison avec la propriété du composant connecteur qui retransmet le paquet de mise à jour traduit des modifications vers le composant DataSet.

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.

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 (Macintosh); $(LocalData)/Classes (Windows)
- .

Le point (.) indique le répertoire actif. Flash recherche les classes ActionScript dans le répertoire courant du fichier FLA.

Les chemins $(UserConfig)/Classes et $(LocalData)/Classes indiquent le répertoire de configuration par utilisateur. Ce répertoire désigne les emplacements suivants :

- Sous Windows, il s'agit du répertoire c:\Documents and Settings*nom d'utilisateur*\Application Data\Macromedia\Flash MX 2004\en\Configuration.
- Sous Macintosh, il s'agit du répertoire *volume*:Users:*nom d'utilisateur*:Library:Application Support: Macromedia:Flash MX 2004:en:configuration.

Les répertoires UserConfig et LocalData 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 ou LocalData.

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 697.

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 674.

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 699.

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 675.

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 674. 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 pour un 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 676.

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 nomClasse: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 sous-composant ScrollBar :

```
function 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 :

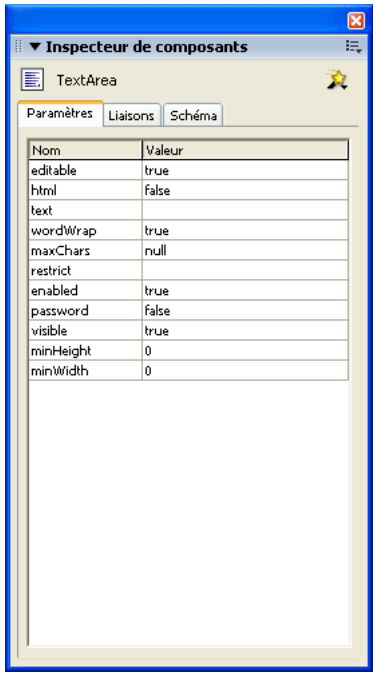
Balise	Description
Inspectable	Définit un attribut fourni aux utilisateurs de composant dans le panneau Inspecteur de composants. Pour plus d'informations, consultez *Inspectable*, page 684.
InspectableList	Identifie les sous-ensembles de paramètres d'inspection à inclure dans l'inspecteur des propriétés. 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 686.
Event	Définit les événements de composant. Pour plus d'informations, consultez *Event*, page 687.
Bindable	Révèle une propriété dans l'onglet Liaisons du panneau Inspecteur de composants. Pour plus d'informations, consultez *Bindable*, page 687.
ChangeEvent	Identifie les événements provoquant la liaison des données. Pour plus d'informations, consultez *ChangeEvent*, page 688.
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 698.

Les sections suivantes décrivent les balises de métadonnées de composant plus en détail.

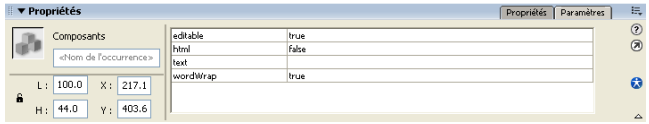
Inspectable

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 688.

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.

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 684). 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.

Evénement UIComponent	Description
`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.
`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 `objEvt` 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 692).

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(objEvt){
  if (objEvt.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 25.

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 687.

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 29.

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.

Fichier	Description
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.
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. Pour plus d'informations, consultez *Ajout d'une icône*, page 698.
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

H

I

L

M

N

O

P

R

S

T

U

W

X